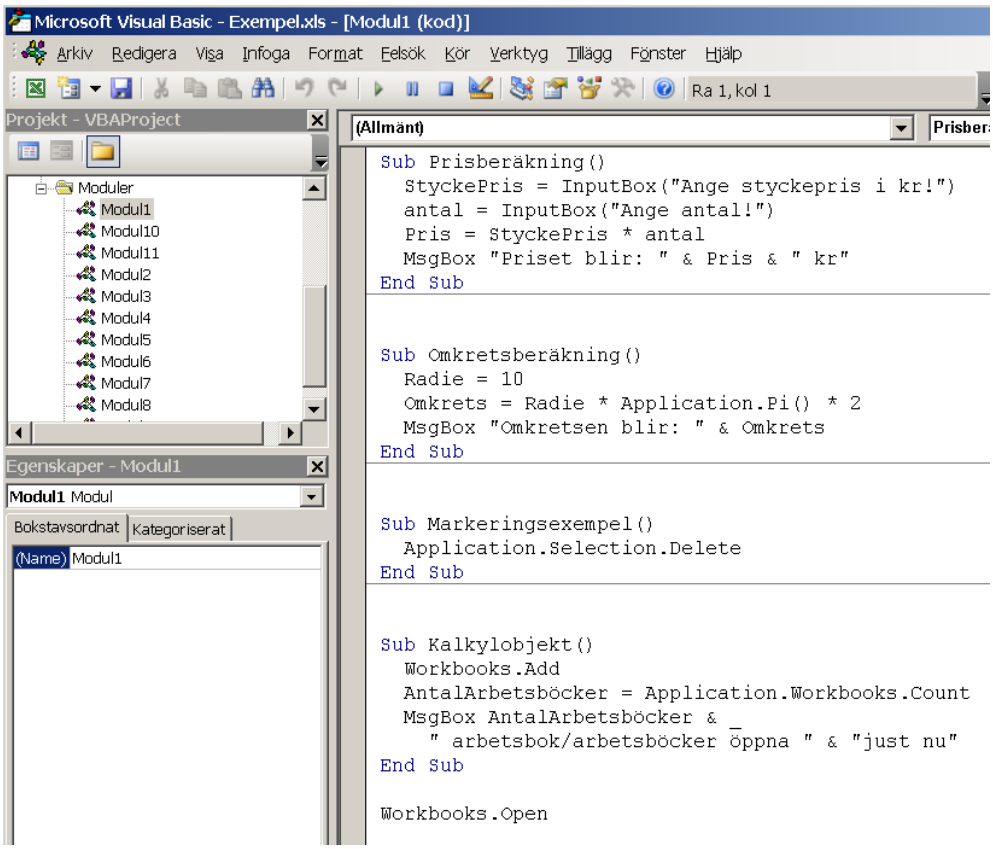
Att använda Visual Basic for Applications (VBA) för Excel

av

Anders Avdic



Version 3.02
2012-06-12

Innehållsförteckning

1 Introduktion

Datoranvändare får allt större möjligheter att utföra allt mer komplicerade funktioner med hjälp av standardprogram. Kalkylprogramet Excel har utvecklats från ett enkelt beräkningsprogram till ett program med vars hjälp man kan utnyttja funktioner för databaser, grafik, diagram, filhantering mm. Trots den ökande funktionaliteten uppstår situationer där den inbyggda funktionaliteten inte räcker till. Då kan ett större eller mindre dataprogram lösa problemet. I detta kompendium visas några enkla lösningar på situationer som kan uppkomma vid utveckling av system.

Programmering kan upplevas som en abstrakt sysselsättning. Till stor del består programmering av att man representerar verkliga storheter med variabler, objekt, metoder, egenskaper och funktioner i programmet. Det kan verka svårt men det är ofta kul. Programmering är en sorts problemlösning.

I detta kompendium behandlas programmering med Visual Basic for Applications (VBA) för Microsoft Excel xp. Från och med version 5.0 av Microsoft Excel finns programspråket VBA tillgängligt för den som vill öka funktionaliteten i sina kalkylsystem. VBA kompletterar och ersätter Excels sk Makrospråk. Excel stöder fortfarande Makrospråket fullt ut men fr.o.m. Excel 5.0 kommer inte nyheter i Excel att stödjas i Makrospråket. Gamla applikationer kan dock köras.

Visual Basic för Excel (VBA) är inte ett av de mest komplexa programspråken men ändå tillräckligt stort för att behöva en omfattande beskrivning om det skulle dokumenteras heltäckande. I detta kompendium är avsikten att ge en introduktion till VBA, inte att ge en heltäckande beskrivning. Kompendiet vänder sig främst till den nyfikna Excelanvändaren som vill öka funktionaliteten i sina applikationer.

2 Grunderna

I detta kapitel behandlas de mest grundläggande VBA-begreppen.

- Moduler, programsatser och kod
- Objekt, egenskaper och och metoder
- Procedurer och funktioner
- Datatyper och variabler

Nämnda begrepp förekommer i i stort sett alla programspråk och är inte på något sätt specifika för VBA. Vad som skiljer VBA för Excel från andra program är just kopplingen till Excel. Hanteringen av kalkylblad förekommer inte i andra programspråk.

2.1 Moduler, kod, programsatser och kommentarer

Ett program sparas i en *modul*. En eller flera moduler utgör ett *projekt*. Ett projekt lagras i en arbetsbok. Åkomst till en modul fås genom *Visual Basic Editor*.
För att kunna skapa eller redigera program i en modul:
- Välj **Verktyg/Makro…/Visual Basic Editor**
eller
- Tryck **Alt+F11**

Ett program utgörs av *kod*. Koden kan ingå i *procedurer* (eller *funktionsprocedurer*).
En procedur inleds med

```
Sub procedurnamn()
```

och avslutas med

```
End Sub
```

Ett *procedurnamn* kan väljas godtyckligt och kan bestå av bokstäver och siffror, men måste inledas med en bokstav. Mellanslag och andra specialtecken som t.ex. punkt (.), komma (,) eller semikolon (;) ä ej tillåtna. Exempel på procedurnamn är Huvudslinga() TaBortPost() och HämtaData3()

En funktion inleds med

```
Function funktionsnamn(argument)
```

och avslutas med

```
End Function
```

För *funktionsnamn* gäller samma regler som för procedurnamn.

I procedurer (och funktioner) finns *programsatser* och/eller *funktioner*. En programsats är en komplett *kodenhet*. En programsats kan sträcka sig över flera rader. Då måste raden som fortsätter avslutas med ett mellanslag och ett understreck (_). En *funktion* är en kodenhet som returnerar ett värde. Excels sk makrospråk består uteslutande av funktioner.
Ett exempel på en programsats är:

```
RadNr = Cells(69, 8).Value + 3
```

Ett exempel på en programsats som sträcker sig över två rader:

```
Värde = CLng(Workbooks(Arbetsbok).Worksheets(Blad).Cells(r1, k1) _
 + Workbooks(Arbetsbok).Worksheets(Blad).Cells(r2, k2))
```

Kod kan förses med *kommentarer*. En kommentar föregås av en apostrof (').
Exempel 1 (kommetaren börjar efter apostrofen):

```
Summa = Summa + Värde  'Ackumulering av värden
```

Exempel 2 (hela raden är en kommentar):

```
'Följande procedur beräknar belopp av typ B
```

2.2 Objekt, egenskaper och metoder

Objekt är det som är föremål för manipulering av något slag t.ex. objektet *Range*.
Objekt har *metoder*. T.ex. har objektet *Range* metoden *Delete* (motsvarar *Ta bort*).
Exempel:

```
Worksheets("Blad1").Range("A1:B3").Delete
```

Objekt har *egenskaper*. T.ex. har objektet *Range* egenskapen *ColumnWidth* (motsvarar *Kolumnbredd*).
Exempel:

```
Worksheets("Blad1").Range("A1").ColumnWidth = 9
```

Några *objekt* i Excel är:

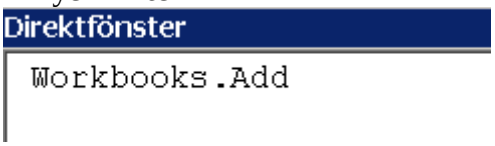
- *Application* (Representerar hela Excel)
- *Workbooks*(Representerar en eller flera arbetsböcker)
- *Worksheets* (Representerar ett eller flera kalkylblad i en arbetsbok)
- *Charts* (Representerar ett eller flera diagram)
- *Range* (Representerar en eller flera celler på ett kalkylblad)
- *PivotTables* (Representerar en eller flera pivottabeller)

- *Comments* (Representerar en eller flera kommentarer)

Några *metoder* för objektet *Range* är:
- *AddComment* Matar in en kommentar i området
- *Replace* Söker upp och ersätter tecken i ett område
- *Select* Markerar angivet område
- *Copy* Kopierar område-objektet till urklipp

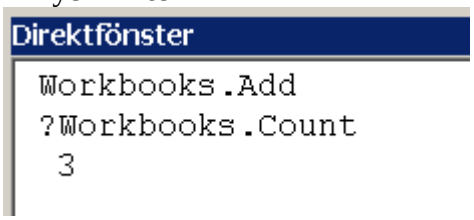
Några *egenskaper* till objektet *Range* är:
- *Width* Returnerar ett områdes bredd i punkter
- *Hidden* Returnerar *True* om dolt
- *Comment* Returnerar områdets kommentar
- *Value* Returnerar värdet i en cell

För att ytterligare visa innebörden av *objekt* och *metoder* visas hur kod kan utföras i direktfönstret:
- Välj **Verktyg/Makro/Visual Basic Editor** (**Alt+F11**)
- Välj **Infoga/Modul**
- Välj **Visa/Direktfönster**
- Aktivera **Direktfönster** (Genom att klicka i det)
- Skriv programsatsen **`Workbooks.Add`** (*objekt.metod*)
- Tryck **Enter**



Nu skapas en ny arbetsbok direkt.
- Tryck **Enter** två gånger till (med markören direkt efter kodraden)
Nu har ytterligare arbetsböcker skapats

För att visa innebörden av *objekt* och *egenskaper* (med direktfönstret):
- Skriv **`?Workbooks.Count`** under den tidigare programsatsen (*objekt.egenskap*)
- Tryck **Enter**



Frågetecknet gör att egenskapens värde visas omgående i fönstret.
Egenskapen *Count* för objektet *Workbooks* hade värdet 3, då tre arbetsböcker var öppna.

För att stänga arbetsböckerna:
- Skriv **`Workbooks.Close`**
- Tryck **Enter**
Böckerna stängs
Close är följaktligen en metod för objektet *Workbooks*.

2.2.1 Mängd

Objektet *Workbooks* ovan är å ena sidan ett objekt med en uppsättning metoder och egenskaper men det är också en *mängd* av objekt. Den engelska termen för mängd är *collection*. Andra exempel på mängder är:

- *Worksheets* Innehåller alla kalkylblad (Worksheet)-objekt i en arbetsbok
- *Dialogs* Innehåller alla dialogblad (Dialog)-objekt i en arbetsbok
- *Charts* Innehåller alla diagram (Chart)-objekt i en arbetsbok
- *Axes* Innehåller alla diagramaxel (Axis)-objekt i ett diagram (Chart)-objekt

2.2.2 Behållare

Vissa objekt innehåller andra objekt och kallas för *behållare*. T.ex. innehåller *Worksheet*-objektet ett antal *Range*-objekt. Ett *Workbooks*-objekt kan innehålla ett *Chart*-objekt etc. Det engelska termen är för behållare är *container*.

2.3 Procedurer (och funktionsprocedurer)

Procedurer kan vara av två sorter, Subprocedurer och funktionsprocedurer. En Subprocedur är ett vanligt program, medan en funktionsprocedur genererar ett värde. En funktionsprocedur kan ta emot noll, ett eller flera argument.

2.4 Variabler och datatyper

Värden i program lagras i *variabler*. Variabler hör till någon *datatyp*. Datatypen beskriver vilken sorts data variabeln kan representera.

2.4.1 Datatyper

Följande datatyper förekommer i VBA:

Datatyp	Beskrivning	Minneskrav	Värdeområde
Boolean		2 byte	*True* eller *False*
Integer	Heltal (ej för stort)	2 byte	-32 768 tom 32 767
Long	Långt heltal	4 byte	-2 147 483 648 tom 2 147 483 647
Single	Flyttal	4 byte	-3,402823E38 tom -1,401298E-45 1,401298E-45 tom 3,402823E38
Double	Flyttal	8 byte	Mycket små och mycket stora tal
Currency	Skalat heltal	8 byte	-922 337 203 685 477,5808 tom 922 337 203 685 477,5807
Date		8 byte	0100-01-01 tom 9999-12-31
String		1 byte/tkn	Beror på
Object		4 byte	Alla objektreferenser (kräver **Set**)
Variant	Default (om ej deklarerat)	Variabel	Alla numeriska värden alla textsträngar
Egendefinierad	med **Type** (se nedan)	Beror på tkn	Beror på grundläggande datatyp
Matris		Variabel	Ordnad mängd element av viss datatyp. Stlk beror på tillg minne

Variabler kan deklareras att tillhöra en viss datatyp i programmet.

Deklarationen sker med programsatsen *Dim*
Format: **Dim** *variabelnamn* [**Som** *typ*]
Exempel:

```
Dim Vikt, Radie, Omkrets
Dim Räknare, RadNr, KolNr As Integer
Dim Kurs As Single, Slutsumma As Double
Dim Fnamn As String, Enamn As String
```

Variabler som inte deklareras kommer att tillhöra datatypen *Variant*

Exempel på egendefinierad datatyp:

```
Type Personuppg
    Namn As String        'String variabel för namn
    Födelsedatum As Date  'Date variabel för födelsedatum
    Kön As Integer        'Integer för kön (0 = kvinna 1 = man)
End Type
```

Man kan tvinga programmeraren att deklarera alla variabler genom att inledningsvis skriva programsatsen `Option Explicit`
Fördelen med att tvingas deklarera variabler är att felsökning förenklas eftersom man inte kan skriva fel variabelnamn utan att det upptäcks.

Man kan se till att `Option Explicit` alltid infogas i kodmoduler genom att:
- Välj **Verktyg/Alternativ...**
- Välj **Redigerare**-fliken
- Kryssmarkera **Variabler måste deklareras**-rutan
- Välj **OK**

2.4.2 Konvertering

Om variabler deklareras som olika datatyper kan det uppstår behov att att konvertera dem i programmet. Det värde som skall konverteras måste förstås vara möjligt att konvertera. Man kan t.ex. inte konvertera string-värdet ”två” till ett integer-värde, däremot kan string-värdet ”2” konverteras. Funktioner för att konvertera är t.ex:
Cint konverterar till Integer
CLng konverterar till Long

För mer information om detta, se *Conversion* i Hjälp om Visual Basic.

3 Inspelning

Ett *makro* är ett litet dataprogram, som utför en viss sekvens av kommandon varje gång man 'kör' det. Man behöver dock inte alltid skriva det för hand, utan man kan ofta helt enkelt spela in när man utför kommandona en gång, ge makrot ett namn och sedan spela upp det när man vill, och då utförs de kommandon man spelade in första gången.
Makrokonstruktion behandlas utförligt längre fram i kompendiet.

Grundmall för makroinspelning
- Tänk igenom noga vilken sekvens av kommandon som skall registreras
- Välj **Verktyg/Makro/Spela in nytt makro...**
- Skriv in ett (beskrivande) namn i **Makronamn**-rutan
- Välj **OK** (**Enter**)
- Utför de kommandon som skall registreras
- Välj **Verktyg/Makro/Stoppa inspelning** (Inspelning avslutas, ordet **Inspelning** försvinner från statusraden)

Grundmall för makrouppspelning
För att köra makro (om *inte* snabbtangent är definierad):
- Välj **Verktyg/Makro/Makron…**
- Välj det makro som skall köras i **Makronamn**-listan
- Välj **Kör**

3.1 Ett enkelt exempel

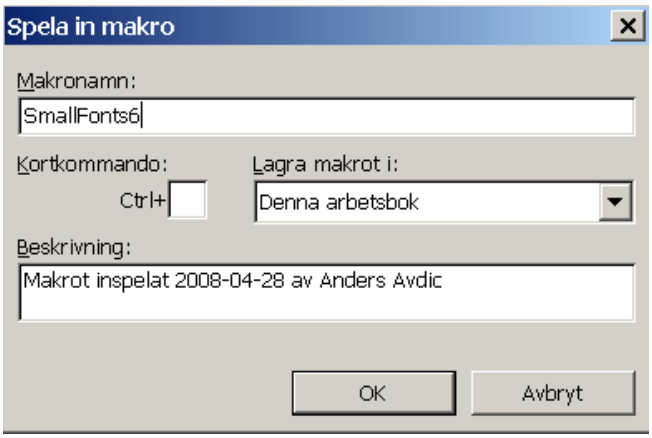
I nedanstående exempel spelas ett makro in som formaterar texten i markerat område till teckensnittet SmallFonts med storlek 6:

	A	B	C	D	E	F	G
1	**Efternamn**	**Förnamn**	**Adress**	**Avd**	**Fack**	**Lön**	**Anställningsår**
2	Ek	Bo	Ekvägen 7	A	Metall	12 500	1994
3	Blom	Siv	Granvägen2	A	Ledarna	30 000	2002
4	Eucalyptusplanta-Jacobs	Gabriella-Victoria	von Reuterkvist-Kuylensti	C	TCO	27 500	1999
5	Gran	Jan	Tallstigen 8	B	TCO	25 000	1986
6	Eucalyptusplanta	Elis	von Reuterkvist-Kuylensti	A	Metall	20 000	1999

Innan inspelningen startar bör man markera det område som makrot skall användas på. Om även markeringen spelas in kommer ju samma område att markeras varje gång.
- Markera **A4:C4**

1	**Efternamn**	**Förnamn**	**Adress**	**Avd**	**Fack**	**Lön**	**Anställningsår**
2	Ek	Bo	Ekvägen 7	A	Metall	12 500	1994
3	Blom	Siv	Granvägen2	A	Ledarna	30 000	2002
4	Eucalyptusplanta-Jacobs	Gabriella-Victoria	von Reuterkvist-Kuylensti	C	TCO	27 500	1999
5	Gran	Jan	Tallstigen 8	B	TCO	25 000	1986
6	Eucalyptusplanta	Elis	von Reuterkvist-Kuylensti	A	Metall	20 000	1999

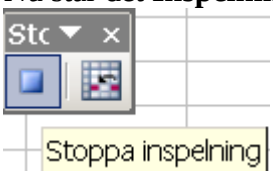
För att börja inspelningen:
- Välj **Verktyg/Makro/Spela in nytt makro...**
- Skriv **SmallFonts6** i Makronamn-rutan



Om makrot används ofta och kan det ta onödigt med tid att köra det via Verktyg/Makro/Kör-menyn. Det finns flera sätt att lösa dett bl.a. genom att definiera en snabbtangent.
- Skriv **s** i **Kortkommando**-rutan

- Klicka **OK**

Nu står det **Inspelning** på Status-raden och **Stoppa inspelning**-knappen placeras på skärmen.



- Välj **Format/Celler.../Tecken**
- Välj **Tecken**-fliken
- Välj **Small Fonts** i **Teckensnitt**-listan
- Välj **6** i **Storlek**-listan

- Välj **OK** (**Enter**)
- Klicka på **Stoppa inspelning**-knappen

eller

- Välj **Verktyg/Makro/Stoppa inspelning**

Inspelning avslutas, ordet **Inspelning** försvinner från statusraden.

3.2 Uppspelning av makro

För att spela upp makrot via *Verktyg/Makro*:
- Markera det område som skall formateras till Small Fonts, storlek 6
- Välj **Verktyg/Makro /Makron...**
- Välj **SmallFonts6** i **Makronamn** -listan
- Välj **Kör**

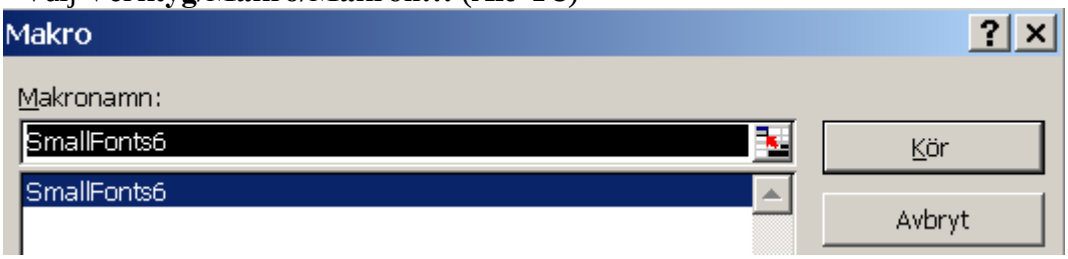
För att spela upp makrot med *snabbtangent*:
- Se till att makrobladet som innehåller makrot är öppet
- Tryck **Ctrl+s**

	A	B	C	D	E	F	G
1	**Efternamn**	**Förnamn**	**Adress**	**Avd**	**Fack**	**Lön**	**Anställningsår**
2	Ek	Bo	Ekvägen 7	A	Metall	12 500	1994
3	Blom	Siv	Granvägen2	A	Ledarna	30 000	2002
4	Eucalyptusplanta-Jacobsson	Gabriella-Victoria	von Reuterkvist-Kuylenstiernas gata 348	C	TCO	27 500	1999
5	Gran	Jan	Tallstigen 8	B	TCO	25 000	1986
6	Eucalyptusplanta	Elis	von Reuterkvist-Kuylenstiernas gata 348	A	Metall	20 000	1999

3.3 **Redigera makro**

Redigerar makron gör man i Visual Basic Editor.
För att starta *Visual Basic Editor*:
- Välj **Verktyg/Makro/Visual Basic Editor** (**Alt+F11**)

För att ett makro redigerbart:
- Välj **Verktyg/Makro/Makron…** (**Alt+F8**)



- Välj önskat makro t.ex. **SmallFonts6**
- Välj **Redigera**

Nu startas Visual Basic Editor
Så här ser makrot ut:

```
Sub SmallFonts6()
'
' SmallFonts6 Makro
' Makrot inspelat 2008-04-28 av Anders Avdic
'

'
    With Selection.Font
        .Name = "Small Fonts"
        .FontStyle = "Normal"
        .Size = 6
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
End Sub
```

Då makrotext redigeras används i Windows vanliga redigeringskommandon.
För att ta bort onödiga rader
- Markera raderna fr.o.m. **`.Strikethrough = False`** t.o.m **`.ColorIndex = xlAutomatic`**
- Tryck **Delete**
- Ta på samma sätt bort rader i början av makrot som inte innehåller något eller bara innehåller en apostrof (’)

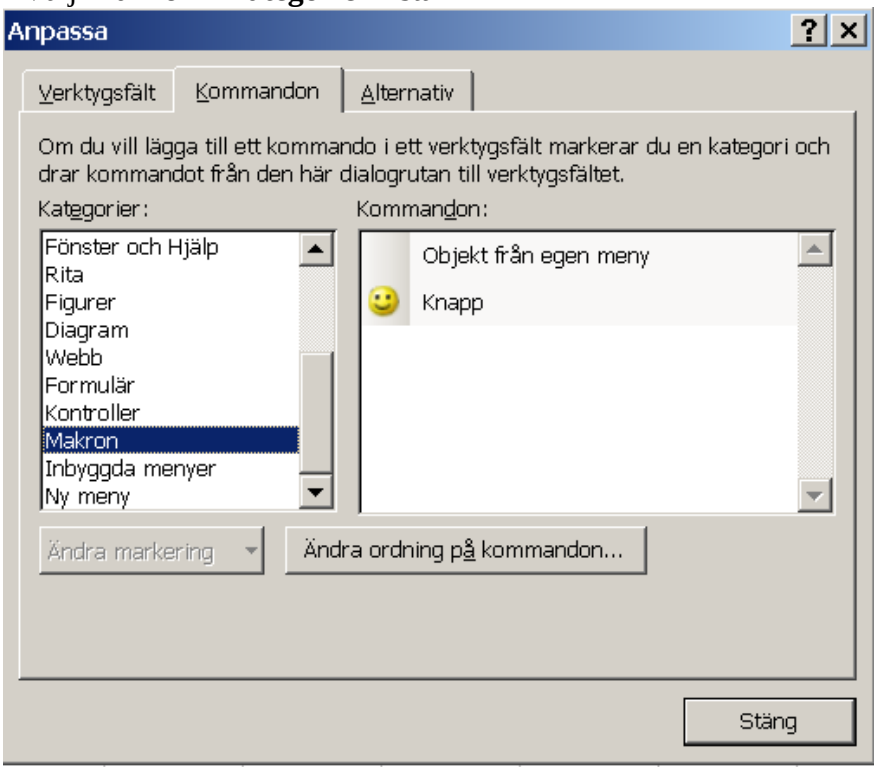
Nu ser makrot ut så här:

```
Sub SmallFonts6()
' Makrot inspelat 2008-04-28 av Anders Avdic
    With Selection.Font
        .Name = "Small Fonts"
        .FontStyle = "Normal"
        .Size = 6
    End With
End Sub
```

3.4 **Koppla makro till knapp**

Makrot kan kopplas till knappar på verktygsfältet eller på kalkylbladet. Om makrot kopplas till en knapp på verktygsfältet kan det användas på flera arbetssböcker. Den arbetsbok som innehåller makrot öppnas automatiskt då man klickar på knappen. Ett makro som kan användas på flera arbetsböcker kan lämpligen sparas i arbetsboken EGNA.XLS för att vara ständigt tillgängligt.

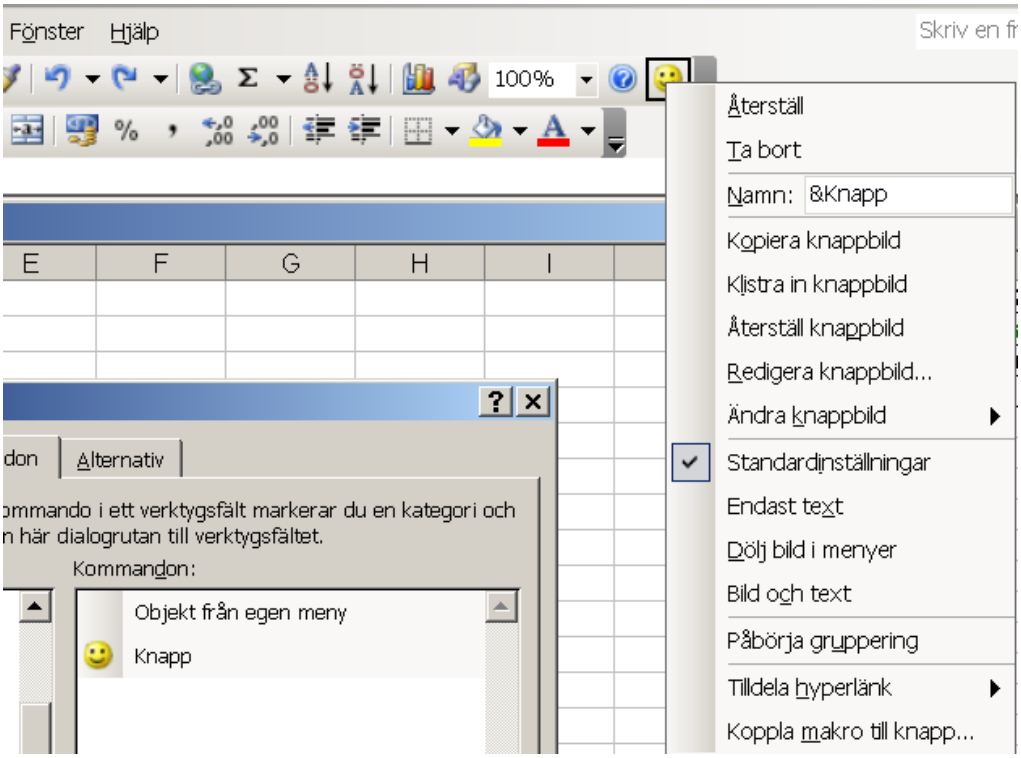
För att koppla makrot till en egen knapp på verktygsfältet:
- Välj **Visa/Verktygsfält/Anpassa…**
- Välj Fliken **Kommandon**
- Välj **Makron** i **Kategorier**-listan



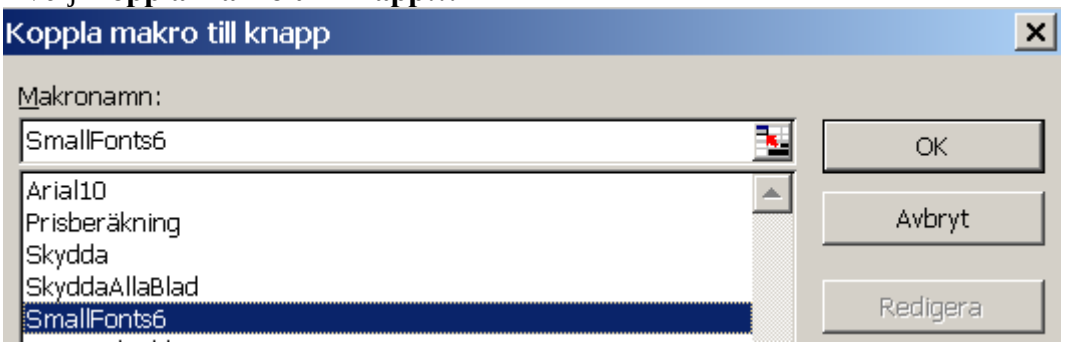
- Dra knappen i **Kommandon**-listan till ett verktygsfält

För att koppla knappen till ett makro:
OBS: Låt Anpassa-rutan vara framme medan knappen kopplas till ett makro!
- Klicka med högerknappen på den nya knappen

Följande meny visas:

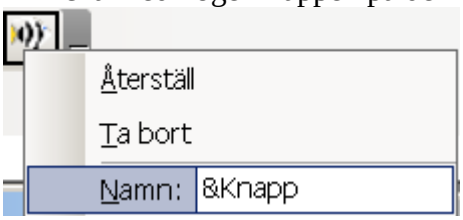


- Välj **Koppla makro till knapp…**

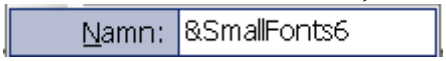


- Välj ett makro, t.ex. **SmallFonts6** i **Makronamn**-listan
- Klicka **OK**

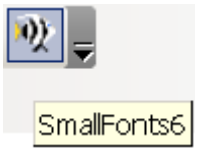
För att ge knappen en beskrivning
- Klicka med högerknappen på den nya knappen



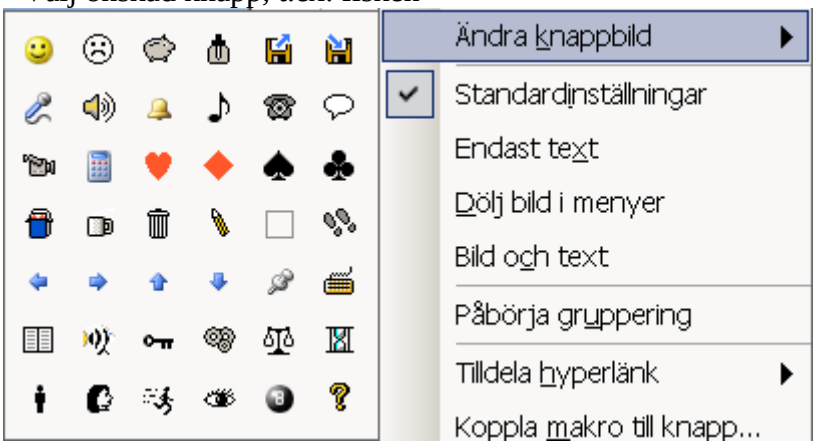
- Skiv ett beskrivande namn, t.ex. SmallFonts6



Beskivningen syns när man hålle muspekaen ovanfö knappen.



För att byta utseende på knappen:
- Klicka med högerknappen på den nya knappen
- Välj **Ändra knappbild**
- Välj önskad knapp, t.ex. fisken



- Välj **Stäng**-knappen i dialogrutan **Anpassa**

Nu kan det se ut så här på verktygsfältsområdet:

*** Använda makro med knapp**

För att köra makrot:
- Markera ett område som skall formateras
- Klicka på Fisk-knappen

3.4.1 Övning: Spela in makro och koppla till knapp

Spela in ett makro som formaterar markerad text till teckensnittet Arial storlek 10. (Dvs tillbaka till standard.)
Redigera bort onödiga rader ur makrot.
Koppla makrot till en knapp på verktygsfältet.

3.5 Redigera och skriva makrokod

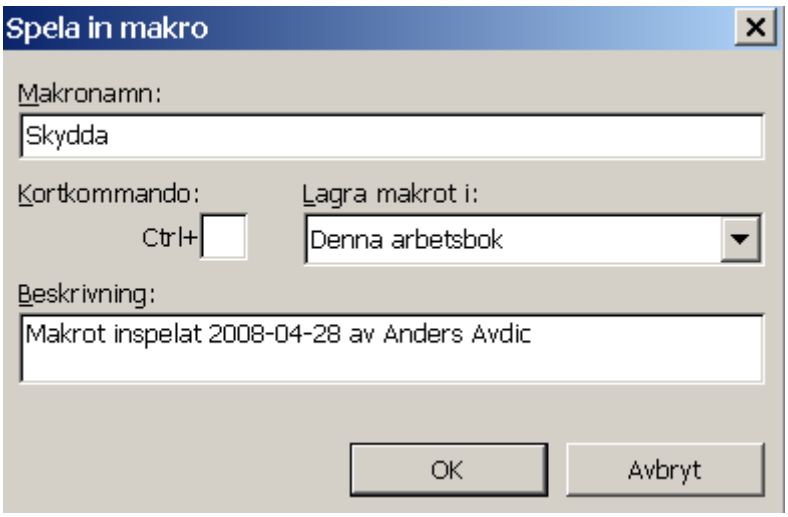
Inspelad kod kan ibland behöva redigeras för att sådant som inte behövs inte skall ta plats. Ett annat skäl att redigera kod är att man kan spela in vissa sekvenser för att få den rätta syntaxen på någon kodrad.

3.5.1 Ta bort onödig kod

Ett användbart makro, om det är kopplat till en knapp, är ett makro som skyddar alla blad i en arbetsbok. Tillsammans med ett makro som tar bort skyddet på arbetsböckerna kan en del onödigt arbete undvikas.

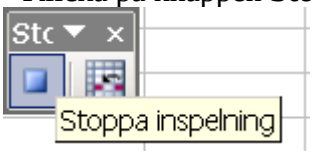
Först skapas ett makro som skyddar ett blad utan arbetsbok.

- Välj **Verktyg/Makro/Spela in nytt makro…**
- Ändra makronamn till **Skydda**
- Välj att spara makrot i **Arbetsboken Egna makron**
- **OK**



Följande skall spelas in:
- Välj **Verktyg/Skydd/Skydda blad…**
- **OK**

För att avsluta inspelningen:
- Klicka på knappen **Stoppa inspelning**



För att redigera makrot:
- Välj **Verktyg/Makro/Makron…**
- Markera makrot **Skydda**
- Välj **Redigera**

Makrot ser ut som följer:

```
Sub Skydda()
'
' Skydda Makro
' Makrot inspelat 2008-04-28 av Anders Avdic
'

'
  ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True
End Sub
```

- Redigera makrot så att det ser ut som följer:

```
Sub Skydda()
  ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True
End Sub
```

- Spara
- Koppla till en knapp med en pil ner  (se ovan)

3.5.2 Skriva kod manuellt

Exemplet utgår från ovanstående inspelade makro Skydda.

För att göra ett makro som skyddar alla blad i en arbetsbok
- Ta fram makrot enligt ovan
    - Skriv manuellt ett makro under det föregående enligt nedan

```
Sub SkyddaAllaBlad()
Dim i, antal As Integer  'Deklaration av tv heltalsvariabler
i = 1                    'Variabeln i sätts till 1
antal = Sheets.Count     'Variabeln antal sätts till antal blad i aktuell bok
While i <= antal         'För varje blad…
  Sheets(i).Select       'Aktivera blad med ordningsnummer i
  Skydda     '**Rutin** Anropar det nyligen inspelade makrot Skydda
  i = i + 1              'Variabeln i räknas upp med 1 (nästa blad)
Wend                     'Här vänder slingan så länge det finns blad
Sheets(1).Select         'Aktivera det första bladet i boken
Range("a1").Select       'Markera cell A1
End Sub
```

- Spara
- Koppla till en knapp 

Hela exemplet Skydda blad finns i bilaga på sid **Fel! Bokmärket är inte definierat.**.

3.5.3 Övning: Redigera och skriva makrokod

Spela in ett makro som tar bort skyddet på det aktiva bladet utan lösen ord:

```
Sub TaBortSkydd()
    ActiveSheet.Unprotect
End Sub
```

Koppla till en knapp med en pil upp 

Skriv ett makro till som anropar detta makro och tar bort skyddet på alla blad i den aktiva boken:

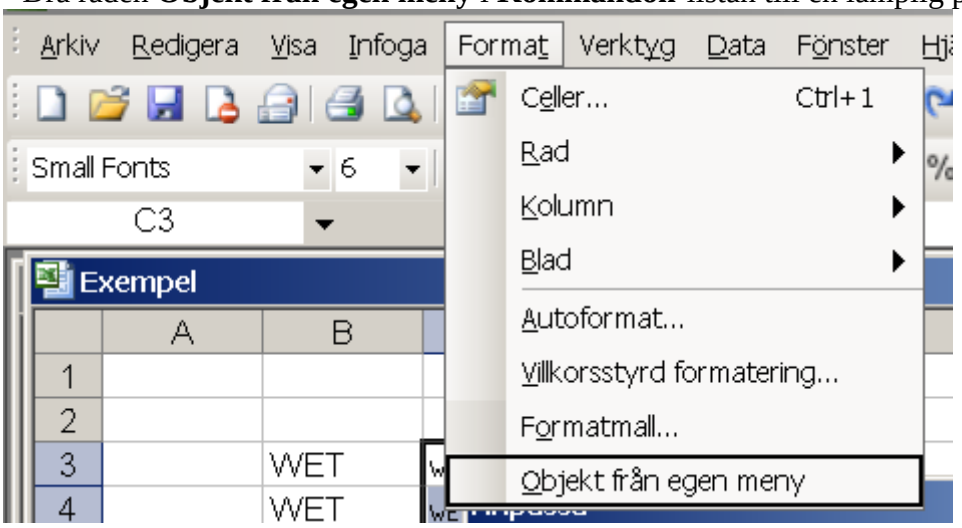
```
Sub TaBortSkyddAllaBlad()
Dim i, antal As Integer
'i = 3 'Om böcker med dolda blad
i = 1
antal = Sheets.Count
```

```
While i <= antal
  Sheets(i).Select
  TaBortSkydd     '**Rutin**
  i = i + 1
Wend
Sheets(1).Select
Range("a1").Select
End Sub
```

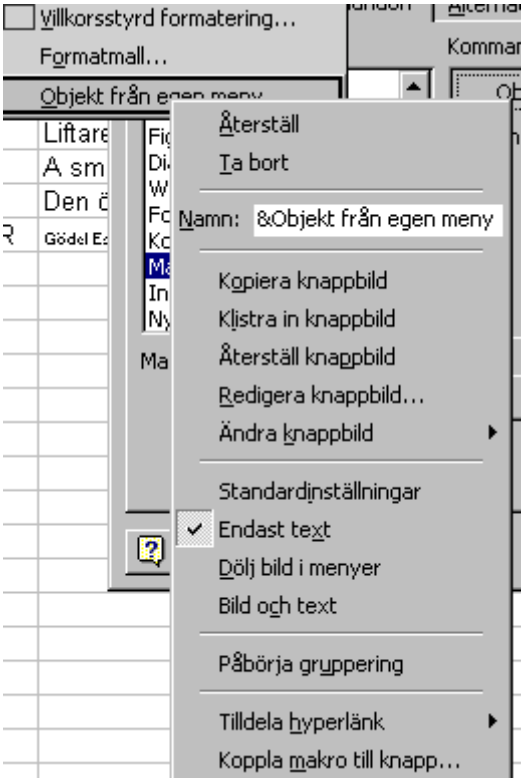
Koppla till en knapp 

3.6 Koppla makro till meny

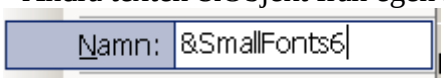
För att koppla ett makro till en befintlig meny:
- Välj **Visa/Verktygsfält/Anpassa…**
- Välj Fliken **Kommandon**
- Välj **Makron** i **Kategori**-listan
- Dra raden **Objekt från egen meny** i **Kommandon**-listan till en lämplig plats i en lämplig meny



- Klicka med högerknappen på det menyval som just lagts in



- Välj **Koppla makro till knapp…** enligt samma arbetsgång som ovan
- Klicka med högerknappen igen på det menyval som just lagts in
- Ändra texten &Objekt från egen meny till lämplig text, t.ex. **SmallFonts6**



- Välj **Stäng**-knappen i dialogrutan **Anpassa**

3.7 Köra makro när bladet öppnas

Om en bok innehåller en procedur med namnet **Auto_open**, så kommer denna procedur att köras när arbetsboken öppnas.
Exempel:

```
Sub Auto_open()
  MsgBox ("Nu är denna bok öppen!")
End Sub
```

Om arbetsboken som öppnas innehåller **Auto_open**-makron, körs de inte när filen öppnas från Visual Basic, dvs om den öppnas av ett annat makro. Om du vill köra makrot Auto_open i ett VBA-program måste metoden **RunAutoMacros** användas.

Ett alternativ:
För att en viss arbetsbok skall öppnas då **Excel** startas kan arbetsboken sparas i en speciell katalog vid namn **XLStart**. **XLStart**-katalogen skapas då **Excel** installeras och kan finnas på olika platser i katalogsystemet beroende på version.

Då ett VBA-makro skapas kan man begära att det skall sparas i arbetsboken Egna.xls som automatiskt placeras i XLStart-katalogen.

3.8 Köra makro när viss tangent trycks ned

Med OnKey kan man få ett visst makro att köras så fort en viss tangent trycks ned. Denna funktion aktiveras genom ett makro som kopplar en viss tangent till ett visst makro.
När nedanstående makro körs så innebär det att makrot KollaHögsta körs så fort Enter (~) trycks ned.

```
Sub Start()
  Application.OnKey "~", "KollaHögsta"
End Sub
```

När nedanstående makro körs så innebär det att makrot KollaHögsta inte körs så fort Enter (~) trycks ned.

```
Sub Slut()
  Application.OnKey "~", ""
End Sub
```

Exemplen ovan är hämtade från ett program som finns i sin helhet i Bilaga på sidan **Fel! Bokmärket är inte definierat.**.

3.8.1 Köra när bladet stängs

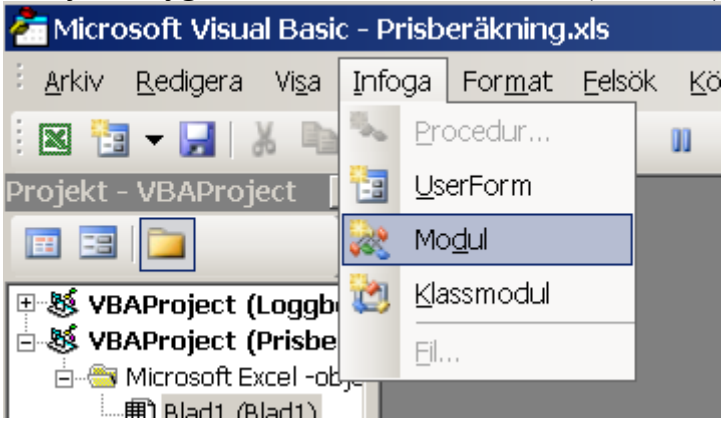
Med makrot *auto_close* kan man få¨ett makro att utföras då en arbetsbok stängs.

```
Sub auto_close()
  Start
End Sub
```

4 Att skriva program

4.1 Ett enkelt exempel

Nedan beskrivs hur ett makro konstrueras som multiplicerar två värden (Antal och styckepris) inmatade från tangentbordet via inmatningsrutor.
För att ta fram ett modulblad:
- Välj **Verktyg/Makro/Visual Basic Editor (eller F11)**



Skriv in följande program i modulen:

```
Sub Prisberäkning()
  StyckePris = InputBox("Ange styckepris i kr!")
```

```
  Antal = InputBox("Ange antal!")
  Pris = StyckePris * Antal
  MsgBox "Priset blir: " & Pris & " kr"
End Sub
```

Kommentarer:
Rad 1: *Sub Prisberäkning()*
Alla makron inleds med **Sub** följt av programnamnet och en parentes
Rad 2: *StyckePris = InputBox("Ange styckepris i kr!")*
StyckePris är en variabel där det inmatade styckeprisvärdet lagras
InputBox är en funktion som gör det möjligt för användaren att mata in ett värde.
Rad 3: *Antal = InputBox ("Ange antal!")*
Samma som rad 2 fast variabeln här har namnet *Antal*
Rad 4: *Pris = StyckePris * Antal*
Här sker en beräkning där resultatet sparas i variabeln *Pris*
Rad 5: *MsgBox "Priset blir: " & Pris & " kr"*
Med funktionen **MsgBox** visas resultatet (*Pris*) tillsammans med en ledtext.
Rad 6: *End Sub*
Programmet avslutas med raden **End Sub**

Spara och stäng Visual Basic Editor

4.1.1 Köra programmet

För att köra programmet ovan:
- Välj **Verktyg/Makro/Makron...**
- Välj *Prisberäkning* i **Makronamn** -listan
- Välj **Kör**

4.1.2 Deklarera datatyper

Genom att deklarera i programmet förekommande variabler lägger man grunden för mer strukturerad programmering. Felsökning underlättas även.
I exemplet ovan skulle programmet inledas med följande variabel deklarationer:

```
Dim StyckePris
Dim Antal
Dim Pris
```

Ett alternativt sätt att skriva samma sak är:

```
Dim StyckePris; Antal; Pris
```

4.1.3 Tilldelning av värden

I ett uttryck av typen `Pris = 4 * 5` tilldelas variabeln till vänster om likhetstecknet (Pris) värdet av beräkningen till höger om likhetstecknet (4*5).

I allmänhet används variabler istället för konstanta värden i program. Då kan uttrycket se ut som följer:

```
Pris = StyckePris * Antal
```

4.2 Makroexempel: Databas och rapport

Kalkylbladet nedan är utgångspunkten för detta exempel. Följande funktioner skall skapas:

- En inmatningsutin
- En Ångra-senaste-inmatning
- En rapport utifrån ett urval, specat med villkorsområde

Eftersom rapporten bygger på funktionen Avancerat filter, så finns följande namn på kalkylbladet:

- Villkor: A3:G4
- Databas: A12:G47 (alla rader i databas + en tomrad)
- Urval: I12:O12

I cell E1 finns en sk databasfunktion som beräknar siumma kostnade på de rader som specificeras av Villkorsområdet.

E1 =DSUMMA(Databas;E3;Villkor)

	A	B	C	D	E	F	G
1	**Reklamationer 2007**				1 989 000		
2	*Villkor*						
3	**Datum**	**Mån**	**Kund**	**Typ**	**Kostn**	**Åtg**	**Avd**
4							
5							
6	*Inmatning*						
7	**Datum**	**Mån**	**Kund**	**Typ**	**Kostn**	**Åtg**	**Avd**
8							
9							
10	*Databas*						
12	**Datum**	**Mån**	**Kund**	**Typ**	**Kostn**	**Åtg**	**Avd**
13	07-12-15	Dec	Wagner	38	210 000	1	A
14	07-12-02	Dec	Smith	99		0	A
15	07-11-08	Nov	Wagner	27	175 000	0	B
16	07-11-04	Nov	Smith	14	0	0	C
17	07-10-15	Okt	Smith	14	84 000	1	B
18	07-10-10	Okt	Pearson	99		1	A

	I	J	K	L	M	N	O
10	*Urval*						
12	**Datum**	**Mån**	**Kund**	**Typ**	**Kostn**	**Åtg**	**Avd**

4.2.1 Rutin för Inmatning

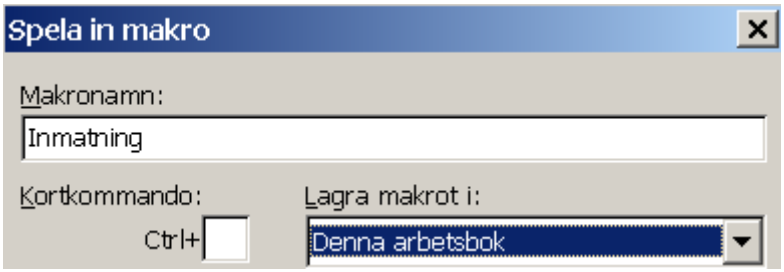
Eftesom databasen är uppbyggd med senast inmatade rad överst så kan in spelning göras där en inmatning på rad 8 kopieas och skjuts in på rad 13.

För att spela in inmatning:

- Skriv följande värden på rad 8

6	*Inmatning*						
7	**Datum**	**Mån**	**Kund**	**Typ**	**Kostn**	**Åtg**	**Avd**
8	2007-12-20	Dec	Smith	14	150 000	0	B

- Välj Makro/Spela in nytt makro…
- Skriv Inmatning i Makronamn-rutan



- OK

Inspelningen börjar. Det står Inspelning på Status-raden och verktygsfältet Inspelning visas.

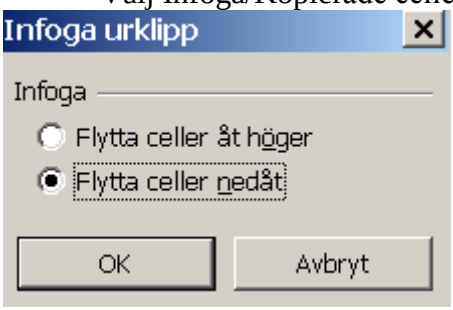
- Markera A8:G8

7	Datum	Mån	Kund	Typ	Kostn	Åtg	Avd
8	2007-12-20	Dec	Smith	14	150 000	0	B

- Välj Redigera/Kopiera (ctrl+c)
- Markera A13:G13

10	*Databas*						
12	Datum	Mån	Kund	Typ	Kostn	Åtg	Avd
13	07-12-15	Dec	Wagner	38	210 000	1	A

- Välj Infoga/Kopierade celler…



- OK

10	*Databas*						
12	Datum	Mån	Kund	Typ	Kostn	Åtg	Avd
13	2007-12-20	Dec	Smith	14	150 000	0	B
14	07-12-15	Dec	Wagner	38	210 000	1	A

För att få rätt format:

- Markera raden under cellerna A14:G14
- Välj Redigera/Kopiera
- Markera A13:G13
- Välj Redigera/Klista in special…/Format/OK

Datum	Mån	Kund	Typ	Kostn	Åtg	Avd
07-12-20	Dec	Smith	14	150 000	0	B

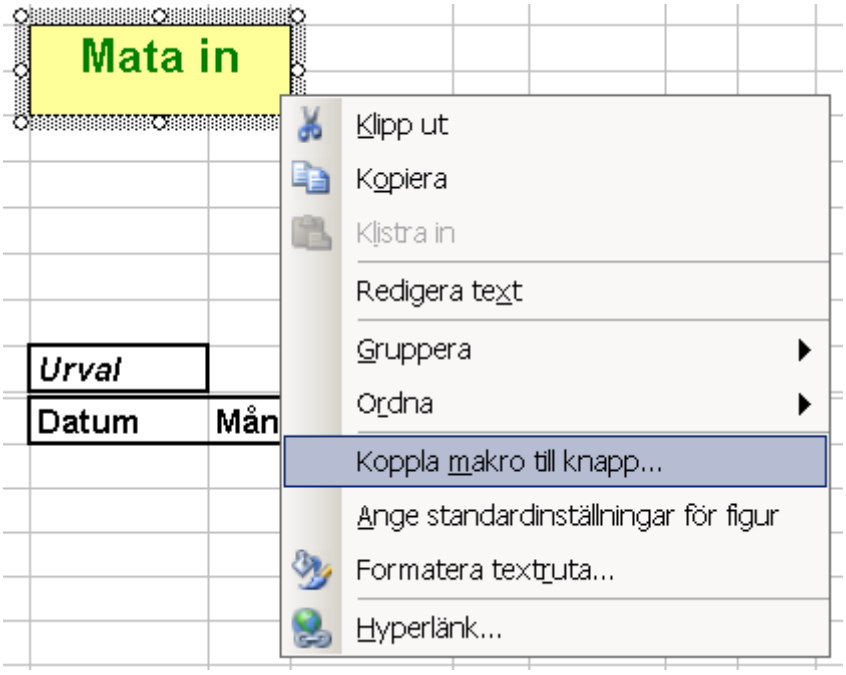
- Markera A8:G8
- Välj Redigera/Radera/Innehåll
- Markera A1
- Tryck Escape (för att tömma Urklipp)
- Stoppa inspelning

Makrot **Inmatning** ser nu ut så här (med några ej helt nödvändiga rader raderade):

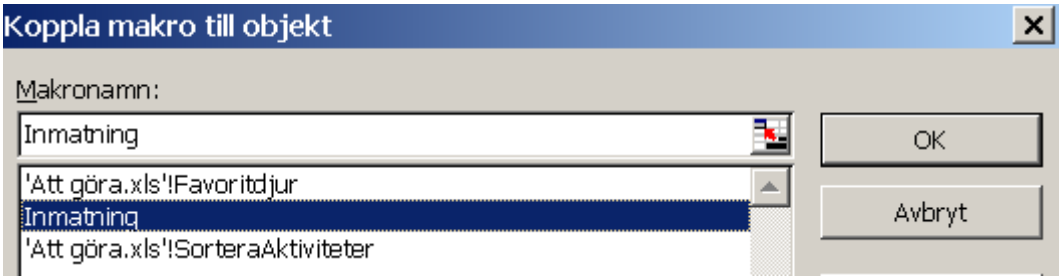
```
Sub Inmatning()
  Range("A8:G8").Select
  Selection.Copy
  Range("A13:G13").Select
  Selection.Insert Shift:=xlDown
  Range("A14:G14").Select
  Application.CutCopyMode = False
  Selection.Copy
  Range("A13:G13").Select
  Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
  Range("A8:G8").Select
  Selection. ClearContents
  Range("A1").Select
  Application.CutCopyMode = False
End Sub
```

För att koppla till knapp

- Ta fram verktygsfältet **Rita**
- Rita en **Textruta**
- Skriv **Mata in** i Textrutan
- Formatera text och ruta
- Högerklicka på rutan
- Välj **Koppla makro till knapp…**



- Välj **Inmatning**/**OK**

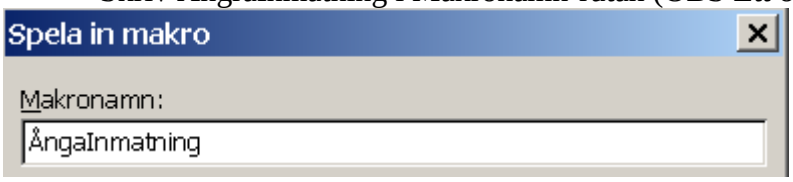


Prova knappen och makrot genom att mata in värden i inmatningsfältet och klicka på knappen.

4.2.2 Rutin för att ångra inmatning

För att spela in ångra inmatning:

- Välj Makro/Spela in nytt makro…
- Skriv ÅngraInmatning i Makronamn-rutan (OBS Ett ord, inga mellanslag)

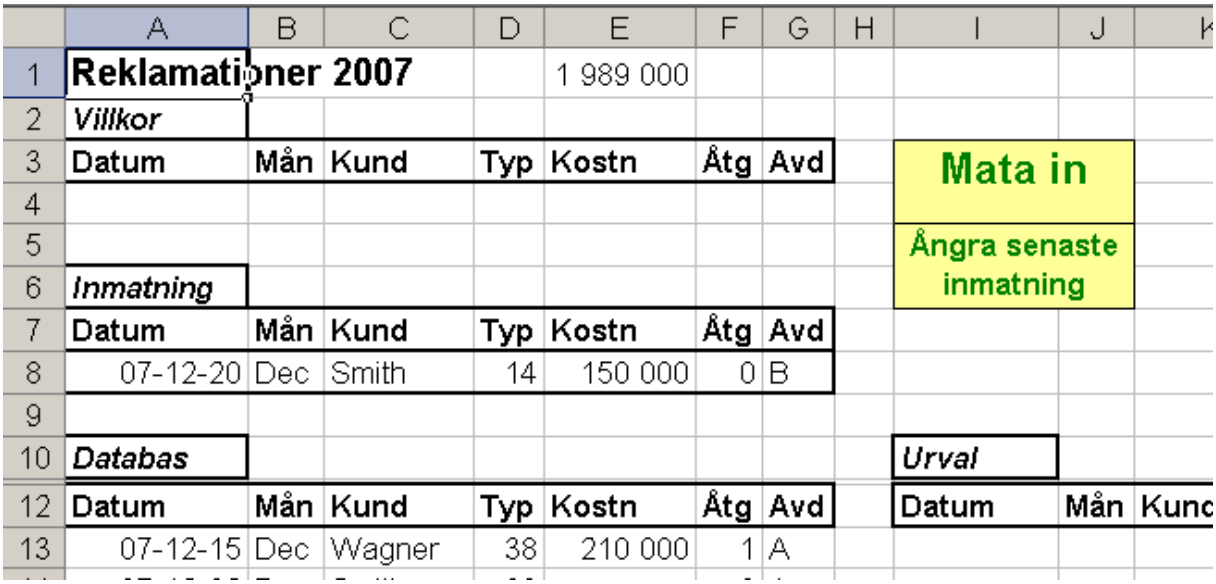


- OK

Inspelningen börjar. Det står Inspelning på Status-raden och verktygsfältet Inspelning visas.

- Markera A13:G13
- Välj Redigera/Kopiera
- Markera A8:G8
- Välj Redigera/Klistra in special/Formler/OK
- Markera A13:G13
- Välj Redigera/Ta bort…
- OK
- Markera A1
- Tryck Escape
- Stoppa inspelning

Gör en knapp med texten Ångra senaste inmatning



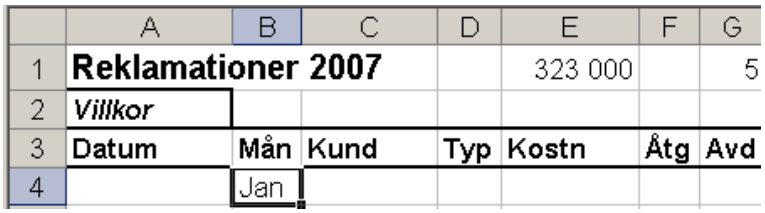
Makrot **ÅngraInmatning** ser efter viss redigering ut så här.

```
Sub ÅngraInmatning()
  Range("A13:G13").Select
  Selection.Copy
  Range("A8:G8").Select
  Selection.PasteSpecial Paste:=xlPasteFormulas, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
  Range("A13:G13").Select
  Application.CutCopyMode = False
  Selection.Delete Shift:=xlUp
  Range("A1").Select
  Application.CutCopyMode = False
End Sub
```

4.2.3 Urval

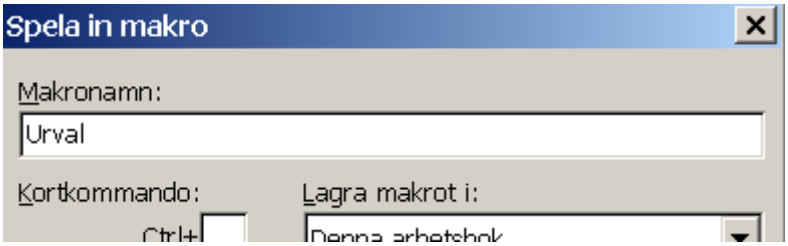
För att spela in ett urval.

Börja med att specifiera villkoren.
Skriv t.ex. in Jan i B4



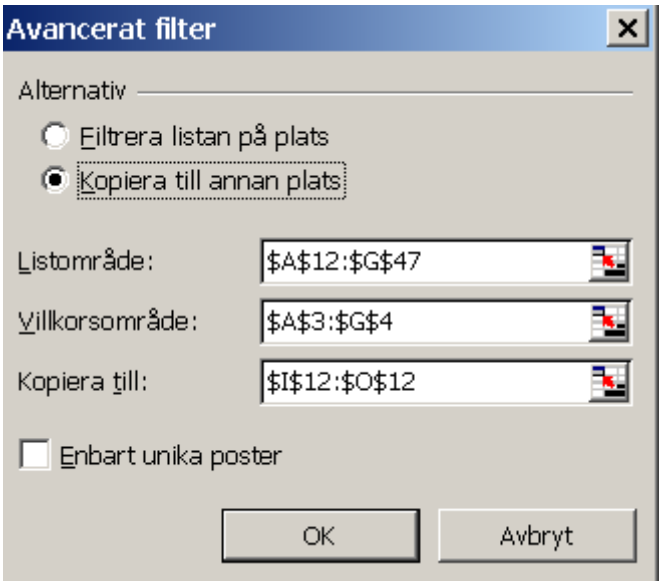
Inspelning:

- Välj **Verktyg/Makro/Spela in nytt makro…**
- Skriv **Urval** i **Namn**-rutan
- **OK**



Nu startar inspelningen

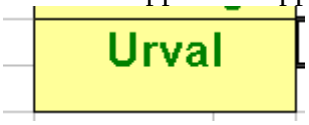
- Välj **Data/Filter/Avancerat filter…**
- Välj alternativ **Kopiera till annan plats**
- **OK**



- Markera **I12**
- Tryck **ctrl+shift+*** (markera aktuellt område)
- Tryck **ctrl+c** (kopiera)
- Klicka på **Nytt…** (ny arbetsbok skapas)

- Tryck **ctlrl+v** (klistra in)
- Markera **A1**
- Avsluta inspelning

Gör en knapp som kopplas till makrot



Nedan visas hur makrot ser ut. Viss inledande rader kan tas bort, dock är det bra att ha datum och namn på den som gjort makrot kvar.

```
Sub Urval()
'
' Urval Makro
' Makrot inspelat 2008-05-11 av Anders Avdic
'

'
    Range("A12:G47").AdvancedFilter Action:=xlFilterCopy,
CriteriaRange:=Range(_
        "A3:G4"), CopyToRange:=Range("I12:O12"), Unique:=False
```

```
    Range("I12").Select
    Selection.CurrentRegion.Select
    Selection.Copy
    Workbooks.Add
    ActiveSheet.Paste
    Range("A1").Select
End Sub
```

4.2.4 Rapport

Anta att en ny kolumn skall skapas i rapporten där kostnaden beräknas om beroende på avdelning. A skall räknas upp med 10 %, B med 20 % och C med 25 %

Aktivera modulen där Urval-makrot finns lagrat.

Skriv:

```
Sub JusteraKostnad()
' Rutin för att infoga kolumn med modifierade kostnader beroende på
avdelning

Const AJust = 0.1
Const BJust = 0.2
Const CJust = 0.25

Dim rad, kol As Integer

Range("H1") = "Just Kostn"
rad = 2: kol = 1

While Cells(rad, kol) <> ""
  If Cells(rad, 7) = "A" Then
    Cells(rad, 8) = Cells(rad, 5) * (1 + AJust)
  ElseIf Cells(rad, 7) = "B" Then
    Cells(rad, 8) = Cells(rad, 5) * (1 + BJust)
  ElseIf Cells(rad, 7) = "C" Then
    Cells(rad, 8) = Cells(rad, 5) * (1 + CJust)
  Else
    'Ingenting
  End If
  rad = rad + 1
Wend
End Sub
```

Spela in sådant som Du tycker behövs föl att snygga till den nya kolumnen, t.ex. kolumn H beddas och när värdena få samma format som kolumn E. Kopiera det som behövs från denn kod till programmet för att får rubriken att bli rätt formaterad.

Låt det första makrot Urval anropa det andra JusteraKostnad så att beräkningen utförs automatiskt.

```
    ActiveSheet.Paste
    Range("A1").Select
    JusteraKostnad  ' Rutinen JusteraKostnad anropas
End Sub
```

5 Goda vanor

5.1 Medan Du kodar

Debugging
Visa variabelvärde

- med cursor
- med Bevakning
- med MsgBox
- Option explicit

5.2 Typografi för ökad läsbarhet av kod

- Kommentera och uppdatera kommentarer
- Indentera
- Använd tomrader i kod för att
- Beskrivande variabler

5.3 Prestanda

- Deklarera variabler med lämplig datatyp. Undvik Variant
- Slå av skärmuppdatering

6 Application, Workbook och Worksheet

Visual Basic är ett generellt programspråk. Det ligger också till grund för VBA (Visual Basic for Application) som finns i Excel, Word och Access. I Excel finns funktioner som är specifika för kalkylbladsmiljön. Några av dem beskrivs nedan.

Objekt	**Samling**	**Beskrivning**
Workbook	Workbooks	Används fö att öppna nya filer och hantera alla filer som för ögonblicket är öppna I Excel.
Worksheet	Worksheets	Används för områden (Range), celler (Cells) och inbäddade object. Det vanligaste objektet.
Application		Controllerar hela Excel. Används fö att avsluta Excel och för att få systeminformation

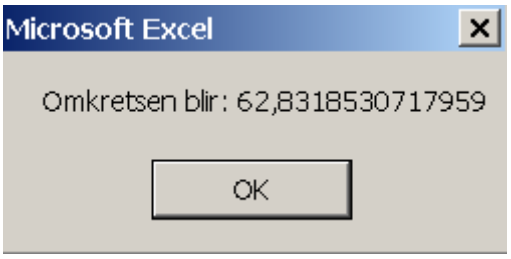
I detta avsnitt skall några för kalkylbladet grundläggande objekt presenteras.

6.1 Application

I en arbetsbok finns speciella objekt. Nedan presenteras några av dem.
Det mest överordnade objektet är objektet *Application*. Kalkylbladsfunktioner i Excel används som metoder till objektet *Application*.
Exempel:

```
Sub Omkretsberäkning()
  Radie = 10
  Omkrets = Radie * Application.Pi() * 2
  MsgBox "Omkretsen blir: " & Omkrets
```

```
End Sub
```
Resultatet blir:



Selection är egenskap som kan användas på objekten *Application* och *Window*.
I nedanstående exempel raderas aktuell markering.

```
Application.Selection.Delete
```

I nedanstående exempel visas antalet rader i ett markerat område. Programmet undersöker om det finns en diskontinuerlig markering. Finns det körs en loop över delområdena i den diskontinuerliga markeringen.

```
AntalOmråden = Markering.Cellområden.Antal
Om AntalOmråden <= 1 Så
  MedRuta "Markeringen innehåller " & _
    Markering.Rader.Antal & " rader."
Annars
  För i = 1 Till AntalOmråden
  MedRuta "Del " & i & " i markeringen innehåller " & _
    Markering.Cellområden(i).Rader.Antal & " rader."
Slut Om
```

6.2 Workbooks

Workbooks och de blad som ingår i en arbetsbok är också objekt med egenskaper och metoder.

Workbooks är en sk ***samling***, den består alltså av ett antal *Workbook*-objekt.
Arbetsböcker kan dels vara ett objekt men även en metod. I nedanstående exempel används först metoden *Add* för att öppna en ny arbetsbok sedan anges variabeln *AntalArbetsböcker* till antalet öppna arbetsböcker. Antal (i uttrycket *`Application.Workbooks.Count`*) är en egenskap hos objektet *Workbooks*.

```
Sub Kalkylobjekt()
  Workbooks.Add
  AntalArbetsböcker = Application.Workbooks.Count
  MsgBox AntalArbetsböcker & _
    " arbetsbok/arbetsböcker öppna " & "just nu"
End Sub
```

Andra metoder till objektet Arbetsböcker är
Close, Open, OpenText
Exempel på metoden Open:

```
Workbooks.Open Filename:="Lager.xls", password:= "passord"
```

6.3 Worksheets m fl

Några så kallade samlingar är Worksheets, Dialogs, Diagram.Nodes och Modules.
Dessa kan t ex användas enligt följande:

```
Worksheets("Blad1").Activate
```

```
MsgBox "Det finns " & ActiveWorkbook.Worksheets.Count & " _
kalkylblad i denna arbetsbok."
```

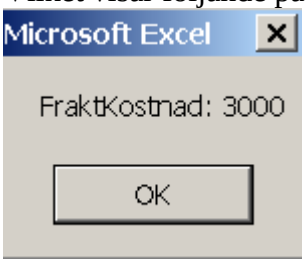
Utifrån följande exempel hämtas viss fraktkostnad

	A	B
1	**Ort**	**Kostnad**
2	Stockholm	5 000
3	Skara	3 000
4	Örebro	500

```
Sub BeräkningFraktkostnad()
Fraktkostnad = Workbooks("EXEMPEL.XLS"). _
    Worksheets("Blad2").Cells(3, 2)
  MsgBox "FraktKostnad: " & Fraktkostnad
End Sub
```

I exemplet tilldelas variabeln Fraktkostnad värdet i cell B3 (Cells(3,2).

Vilket visar följande på bildskärmen:



6.4 Workbook

Workbook har en mängd egenskaper och metoder. Några av dem exemplifieras nedan.

I nedanstående exempel visas namnet och det fullständiga namnet på aktiv arbetsbok.

```
Sub ExempelNamn()
With ActiveWorkbook
  MsgBox "Detta är arbetsboken " & .Name
  MsgBox "Dess fullständiga namn är " & .Path
End With
End Sub
```

Egenskapen *HasPassword* kan anta värdet *True* eller *False*. I nedanstående exempel testas den aktiva arbetsboken har ett lösenord.

```
Sub PassordFinns()
If ActiveWorkbook.HasPassword = True Then
  MsgBox ActiveWorkbook.Name & " är skyddad och kan ej läsas."
Else
  MsgBox "Passord saknas för " & ActiveWorkbook.Name
End If
End Sub
```

6.5 Worksheet/ActiveSheet

Med objektet *Worksheet* hanteras ett visst kalkylblad.
I nedanstående exempel skrivs det aktiva bladet ut med metoden *Print*.

```
ActiveSheet.PrintOut
```

Här följer ett exempel på tilldelning av värdet till cell E5 på det aktiva bladet.

```
ActiveSheet.Cells(5,5) = "test"
```

6.6 Range

Med objektet *Range* kan delar av kalkylblad hanteras.
Exempel:

```
Range("A1:C3").Select
Range("B2").Activate
```

I nedanstående exempel markeras A1:B3 i det aktiva bladet.

```
Range(Cells(1, 1), Cells(3, 2)).Select
```

Detta sätt att utnyttja område gör det lättare att använda variabler för att adressera delar av kalkylbladet variabelt.

Metoden *Rows* kan användas på objekten *Application*, *Worksheet* och *Range*.
I nedanstående exempel tas den tredje raden bort i aktivt blad.

```
Application.Rows(7).Delete
```

Ett objekt kan vara av en annan objekttyp och därmed ha tillgång till detta objekts egenskaper och metoder. Objektet *Selection* är ett *Range*-objekt. Detta innebär att metoder och egenskaper som gäller för *Range* även gäller för *Selection* och andra *Range* -objekt, som t.ex. *Cells*.

6.7 Hantering av objekt

För att utföra en eller flera programsatser på ett objekt kan *With* användas. I följande exempel formateras innehållet i en cell. Objektet är här *Selection*.

```
Sub WithTest()
  Workbooks.Add
  ActiveCell.Value = "Budget 2009"
  With Selection.Font
    .Name = "Courier"
    .Size = 16
    .Bold = True
  End With
End Sub
```

6.7.1 Tilldelning

Tilldelning kan ske till kalkylbladsceller med hjälp av Range

```
Sub Tilldelning()
  Worksheets("Blad1").Range("A1") = "Namn"
  Worksheets("Blad1").Range ("A2") = "Blom"
  Worksheets("Blad1").Range ("A3") = "Holm"
End Sub
```

eller om cellen har ett namn, t.ex. MyCell och värdet finns i en variabel, t.ex. YourValue.

```
  Range("MyCell").Value = YourValue
```

6.7.2 Formatering

Kalkylbladsceller kan formateras.

```
Sub Formatering1()
  Worksheets("Blad1").Range("A1").Font.Bold = True
End Sub
```

I nedanstående exemplet anges teckensnittet i den aktiva cellen till 12 punkter fet, kursiv stil.

```
Sub Formatering2()
With ActiveCell.Font
```

```
  .Size = 14
  .Bold = True
  .Italic = True
End With
End Sub
```

7 Programstyrning

Programstyrning kan innebära att en viss sekvens utförs flera gånger (*loop* eller *slinga*) eller att olika sekvenser utförs beroende på utvärdering av ett *villkor* (*selektion*).
Loopar kan utföras på flera sätt. En *antalsloop* utförs ett visst antal gånger medan en *villkorsloop* utförs så länge ett villkor är uppfyllt.

7.1 Selection: If...Then...Else...End If

Med *If...Then...Else...End If* kan olika programsatser utföras beroende på om ett villkor är sant eller falskt.
I nedanstående exempel utvärderas ett värde som matas in från tangentbordet.

```
Dim Ålder
Sub OmTest1()
  Ålder = InputBox("Hur gammal är Du?")
  If Ålder >= 18 Then
    MsgBox "Du är myndig"
  End If
End Sub
```

Villkoret i exemplet är **Ålder >= 18** Detta utvärderas och om det inmatade värdet är 18 eller större skrivs texten *Du är myndig* ut, annars händer inget.

I nästa exempel skall texten *Du är inte myndig* skrivas ut om värdet är mindre än 18.

```
Dim Ålder
Sub OmTest2()
  Ålder = InputBox("Hur gammal är Du?")
  If Ålder >= 18 Then
    MsgBox "Du är myndig"
  Else
    MsgBox "Du är inte myndig"
  End If
End Sub
```

7.1.1 Nästning av If-satser

```
Dim Ålder
Sub OmTest()
  Ålder = InputBox("Hur gammal är Du?")
  If Ålder >= 18 Then
    If Ålder >= 65 Then
      MsgBox "Du är pensionär"
    Else
      MsgBox "Du är myndig"
    End If
  Else
    MsgBox "Du är inte myndig"
 End If
End Sub
```

7.2 Slingor

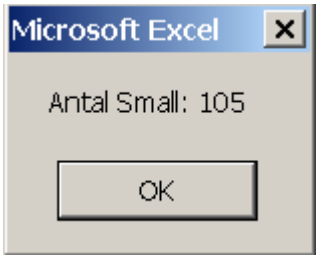
I exemplen nedan förutsätts att det på kalkylbladet Blad1 finns följande matris

	A	B	C	D
1		**Röd**	**Blå**	**Grön**
2	**Small**	20	55	30
3	**Medium**	30	10	15
4	**Large**	25	50	35

7.2.1 For...Next

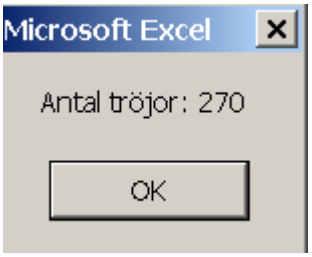
I exempel 1 summeras samtliga Small-tröjor:

```
Dim Ko Som Heltal
Dim TröjSumma
Sub SummeringSmall()
  TröjSumma = 0
  För Ko = 2 Till 4
    TröjSumma = TröjSumma + Kalkylbladlista("Blad1") _
    .Celler(2; Ko)
  Nästa
  MedRuta "Antal Small: " & TröjSumma
Slut Sub
```



I exempel 2 summeras samtliga siffror

```
Dim Ra; Ko Som Heltal
Dim TröjSumma
Sub SummeringAlla()
  TröjSumma = 0
  För Ko = 2 Till 4
    För Ra = 2 Till 4
      TröjSumma = TröjSumma + Kalkylbladlista("Blad1") _
      .Celler(Ra; Ko)
    Nästa
  Nästa
  MedRuta "Antal tröjor: " & TröjSumma
Slut Sub
```



7.2.2 While…Wend

Med While…Wend skapas en sk villkorsloop som utförs så länge ett villkor är uppfyllt.

I exemplen nedan förutsätts att det på kalkylbladet Blad1 finns följande matris

	A	B	C	D
1		Röd	Blå	Grön
2	Small	20	55	30
3	Medium	30	10	15
4	Large	25	50	35

Beräkningen sker genom markering av cell varefter programmet undersöker om där finns något värde. Om så är fallet summeras värdet till en räknare (TröjSumma). När en tom cell markeras avbryts loopen.

```
Dim Ra, Ko
Dim TröjSumma
Sub MedanTestSmall()
  Ra = 2
  Ko = 2
  TröjSumma = 0
  Worksheets("Blad1").Activate
  Cells(Ra, Ko).Select
  While Not IsEmpty(ActiveCell)
    TröjSumma = TröjSumma + ActiveCell.Value
    Ko = Ko + 1
    Cells(Ra, Ko).Select
  Wend
  MsgBox "Antal Small: " & TröjSumma
End Sub
```

Om programmet inte körs från den aktuella arbetsboken så skall programsatsen

```
Worksheets("Blad1").Activate
```

skrivas

```
Workbooks("EXEMPEL.XLS"). Worksheets("Blad1").Activate
```

7.2.3 Bearbetning utan att markera celler

Om Området som skall undersökas är stort går det fortare att utföra bearbetningen utan att markera celler. Följande program fungerar på det sättet:

```
Dim Ra, Ko
Dim TröjSumma
Sub MedanTestSmallEjMarkering()
  Ra = 2
  Ko = 2
  TröjSumma = 0
  Worksheets("Blad1").Activate
  While Not IsEmpty(Cells(Ra, Ko))
    TröjSumma = TröjSumma + Cells(Ra, Ko).Value
    Ko = Ko + 1
  Wend
  MsgBox "Antal Small: " & TröjSumma
End Sub
```

I följande exempel beräknas totalsumman på alla värden genom att infoga en *Medan*-loop i en annan *Medan*-loop.

```
Dim Ra, Ko
Dim TröjSumma
Sub MedanTestDubbelLoop()
  Ra = 2
  Ko = 2
  TröjSumma = 0
  Worksheets("Blad1").Activate
  While Not IsEmpty(Cells(Ra, Ko))
    While Not IsEmpty(Cells(Ra, Ko))
      TröjSumma = TröjSumma + Cells(Ra, Ko).Value
      Ko = Ko + 1
```

```
    Wend
    Ko = 2
    Ra = Ra + 1
  Wend
  MsgBox "Antal tröjor: " & TröjSumma
End Sub
```

7.2.4 Nästning While…Wend

Nedanstående exempel undersöker hur många värden i ett område som överstiger ett visst fastställt värde (40).

```
Sub Kontroll()
Dim Ra, Ko, Räknare As Integer
Dim KontrollVärde
  Ra = 2
  Ko = 2
  Räknare = 0
  KontrollVärde = 40
  While Not IsEmpty(Cells(Ra, Ko))
    While Not IsEmpty(Cells(Ra, Ko))
      If Cells(Ra, Ko).Value > KontrollVärde Then
        Räknare = Räknare + 1
      End If
      Ko = Ko + 1
    Wend
    Ko = 2
    Ra = Ra + 1
  Wend
  MsgBox ("Antal tröjtyper med över " & KontrollVärde & _
" sålda enheter: " & Räknare)
End Sub
```

Om man vill kunna ändra kontrollvärdet vid körningen av programmet
Genom att ändra raden

```
KontrollVärde = 40
```

till

```
KontrollVärde = CDbl(InputBox("Ange kontrollvärde!"))
```

7.2.5 Exempel:

Följande exempel utgår från en tabell som hämtats från ett ekonomiprogram till Excel för vidarebearbetning (se tabell nedan). I kolumnen Belopp innehåller värdena sk. hårda mellanslag, som inte låter sig rensas bort med RENSA eller BYT.UT.
Ett sätt att rensa beloppsvärdena är att gå igenom kolumnen rad för rad och i varje cell spara alla tecken som inte hårda mellanslag. Programmet måste kunna klara även de värden som är numeriska (t.ex. cellerna E8 och E10).

E2 -13481390

	A	B	C	D	E
1	Konto	Org	Kurs/pro	V-gren	Belopp
2	30201	2000	A000	10	-13 481 390,00
3	302732	2013	A011	10	-525 000,00
4	314060	2013	A000	10	-69 529,00
5	314060	2021	A001	10	-11 280,00
6	314060	2031	A000	10	-22 761,00
7	314060	2031	A001	10	-44 370,00
8	314060	2031	A012	10	-200,00
9	314060	2041	A001	10	-14 800,00
10	314060	2051	A001	10	-200,00
11	319999	2013	A000	10	69 529,00
12	333005	2021	A123	10	-40 000,00

Programmet förklaras nedan:

```
Option Explicit
'Variabler:
Dim Rad, Kol, Längd, i As Integer
Dim CellVärde1, CellVärde2
Dim CellVärdeNum As Double
Sub RensaBlanka()
' RensaBlanka Makro
' Makrot inspelat 2002-04-13 av Anders Avdic
Rad = ActiveCell().Row           'Spara radnummer för aktiv cell
Kol = ActiveCell().Column        'Spara kolumnnummer för aktiv cell
While Cells(Rad, Kol) <> ""      'Initiera loop, pågår tills en cell är tom
  CellVärde1 = Cells(Rad, Kol)   'Spara värdet i en variabel
  CellVärde2 = ""                'Nollställ mottagarvariabel
  Längd = Len(Cells(Rad, Kol))    'Spara längden på cellens värde
  For i = 1 To Längd             'Gå igenom cellen tecken för tecken
    If Asc(Mid(CellVärde1, i, 1)) <> 160 Then 'Ta ej med hårt mellanslag
      CellVärde2 = CellVärde2 + Mid(CellVärde1, i, 1) 'Lägg till tecken
    End If
  Next i                         '...nästa tecken i cellen
  CellVärdeNum = CellVärde2      'Gör värdet i mottagarcellen numeriskt
  Cells(Rad, Kol) = CellVärdeNum 'Mata in det nya värdet i cellen
  Rad = Rad + 1                  'Öka radnumret med 1
Wend                             'Pröva villkoret på nästa rad
End Sub                          'Slut på programmet
```

Programmet utgår från att den översta cellen i den kolumn som skall rensas är aktiv.

`Option Explicit` innebär att alla variabler måste deklareras

Raderna med `Dim` är deklarationer av variabler som används i programmet

Programmet inleds med `Sub RensaBlanka()`

`Rad = ActiveCell().Row` och `Kol = ActiveCell().Column` sparar rad- och kolumnnummer på den markerade cellen

`While Cells(Rad, Kol) <> ""` Startar villkorsloopen. Innebär att det som står mellan While och Wend upprepas så länge som `Cells(Rad, Kol) <> ””` dvs så länge det finns något i den cell som behandlas.

`CellVärde1 = Cells(Rad, Kol)` Innebär att det värde som står i den cell som behandlas sparas i variabeln CellVärde1

`CellVärde2 = ""` Innebär nollställning av den variabel som skall ta emot innehållet i CellVärde1, tecken för tecken

`Längd = Len(Cells(Rad, Kol))` Innebär att antalet tecken i den cell som behandlas sparas i variabeln Längd

For `i = 1 To Längd` Innebär starten på en antalsloop som ska gå igenom tecknen i CellVärde1 tecken för tecken. Variabeln i är en räknare som räknas upp med ett för varje varv i loopen.

`If Asc(Mid(CellVärde1, i, 1)) <> 160` Then innebär att man första loopvarvet undersöker om det första tecknet är ett hårt mellanslag (med ASCII-kod 160)

`CellVärde2 = CellVärde2 + Mid(CellVärde1, i, 1)` Om det inte är det läggs tecknet till variabeln CellVärde2. Detta innebär att hårda mellanslag ignoreras. Med funktionen Mid plockas rätt tecken, dvs teckn nr i.

`End If` innebär slutet på villkoret If Asc…

`Next i` inebär att variabeln i räknas upp för att kunna titta på nästa tecken i CellVärde1. Dett pågär tills i är lika med variabeln Längd, dvs en gång för varje tecken i CellVärde1.

`CellVärdeNum = CellVärde2` Gör om värdet i CellVärde2 till ett numeriskt värde tack vare att CellVärdeNum är deklarerat som en Double-variabel

`Cells(Rad, Kol) = CellVärdeNum` Här matas det nya rensade värdet in i den cell som det hämtades från.

`Rad = Rad + 1` Innebär att radvariabeln räknas upp med ett så att Cells(Rad, Kol) skall peka på raden under.

`Wend` avslutar villkorsloopen.

`End Sub` Avslutar programmet.

7.2.6 For Each...Next

For Each...Next är en loop som kan passa bra att använda på kalkylblad då man vill hantera ett område på kalkylbladet.

I nedanstående exempel omvandlas celler med formler till värden i det aktuella området (det område som är markerat då makrot körs).

```
Option Explicit
Dim cell As Range
Sub PasteSpecialValues_example1()
  On Error Resume Next
  For Each cell In Selection
    cell.Formula = cell.Value
  Next cell
End Sub
```

Alternativt:

```
For Each cell In Selection.specialcells(xlvisible)
  cell.Formula = cell.Value
Next cell
```

I nedanstående exempel beräknas tröjförsäljningen (se tabell ovan) med hjälp av *For Each...Next*

```
Dim TröjSumma
Dim TröjCell
Sub SummeringAllaOmråde1()
  TröjSumma = 0
  For Each TröjCell In Worksheets("Blad1").Range("b2:d4")
      TröjSumma = TröjSumma + TröjCell.Value
  Next
  MsgBox "Antal tröjor: " & TröjSumma
End Sub
```

Genom att deklarera en objektvariabel (FörsOmråde) och tilldela den en objektreferens (med hjälp av *Set*) kan skrivsättet förenklas enligt nedan.

```
Dim TröjSumma
Dim TröjCell
Dim FörsOmråde As Object
Sub SummeringAllaOmråde2()
  Set FörsOmråde = Worksheets("Blad1").Range("b2:d4")
  TröjSumma = 0
  For Each TröjCell In FörsOmråde
      TröjSumma = TröjSumma + TröjCell.Value
  Next
  MsgBox "Antal tröjor: " & TröjSumma
End Sub
```

7.3 Spara namn på arbetsböcker och blad

Ibland kan programmet behöva växla mellan arbetsböcker (eller kalkylblad). Om det ena bladet skapas av programmet så vet man inte vilket namn det har. Då kan man använda egenskaperna AktivArbetsbok (eller AktivtBlad).
I nedanstående exempel sparas den aktiva arbetsbokens namn i variabeln AktivBokNamn

```
Sub VäxlaBok1()
Dim AktivBokNamn
  AktivBokNamn = ActiveWorkbook.Name
  MsgBox "Aktiv arbetsbok är " & AktivBokNamn
End Sub
```

7.3.1 Växla mellan arbetsböcker

I nedanstående exempel sparas namnet på den aktiva arbetsboken, varefter en ny arbetsbok skapas. Slutligen aktiveras den ursprungliga arbetsboken.

```
Sub VäxlaBok2()
Dim BokNamn1, BokNamn2
  BokNamn1 = ActiveWorkbook.Name
  Workbooks.Add
  BokNamn2 = ActiveWorkbook.Name
  Workbooks(BokNamn1).Activate
End Sub
```

7.3.2 Exempel

I nedanstående exempel skapas en rapport på ett nytt blad utifrån en tabell med följande utseende:

	A	B	C	D	E
1	**Artikelnr**	**Artikelgrupp**	**Månad**	**Budget**	**Utfall**
2	45-350	Mini	Jan	450	552
3	45-450	Mini	Jan	550	559
4	46-500	Mini	Jan	600	597
5	46-550	Mini	Jan	340	402
6	46-650	Mini	Jan	640	548
7	55-350	Medel	Jan	700	784
8	55-450	Medel	Jan	550	536
9	65-350	Medel	Jan	600	661
10	65-450	Medel	Jan	750	742
11	88-350	Mega	Jan	790	821
12	88-450	Mega	Jan	790	862
13	89-500	Mega	Jan	690	807
14	96-550	Mega	Jan	720	795
15	96-650	Mega	Jan	700	830
16	96-750	Mega	Jan	650	738

Programmet undersöker kvoten mellan Utfall och Budget och jämför den med ett inmatat kontrollvärde. En ny arbetsbok skapas för en rapport. Om kvoten är större än kontrollvärdet flyttas artikelnummer och kvot över till rapporten. Bladet som tabellen ligger på heter BudgetUtfall. Programmet består av fyra procedurer, som anropas av varandra.

```
Option Explicit

Dim BudgetBok, RapportBok, RapportBlad
Dim Ra1, Ko1, Ra2, Ko2 As Integer
Dim KontrollVärde As Single
Dim Kvot As Single

Sub Rapportering()      'Huvudslinga
  Initiering
  Behandling
End Sub

Sub Initiering()
  Ra1 = 2
  Ko1 = 1
  Ra2 = 1
  Ko2 = 1
  KontrollVärde = CDbl(InputBox("Ange kontrollvärde!"))
  BudgetBok = ActiveWorkbook.Name
  Worksheets("BudgetUtfall").Activate
  Workbooks.Add
  RapportBok = ActiveWorkbook.Name
  ActiveSheet.Cells(Ra2, Ko2).Value = "Art nr"
  ActiveSheet.Cells(Ra2, 2).Value = "Kvot"
  Workbooks(BudgetBok).Activate
End Sub

Sub Behandling()
  Dim Budget, Utfall As Integer
  While Not IsEmpty(ActiveSheet.Cells(Ra1, Ko1))
    Budget = ActiveSheet.Cells(Ra1, 4)
    Utfall = ActiveSheet.Cells(Ra1, 5)
    Kvot = Utfall / Budget
    If Kvot >= KontrollVärde Then
      FlyttaRad
```

```
    End If
    Ra1 = Ra1 + 1
  Wend
  Workbooks(RapportBok).Activate
End Sub

Sub FlyttaRad()
  Dim ArtNr
  ArtNr = ActiveSheet.Cells(Ra1, 1)
  Workbooks(RapportBok).Activate
  Ra2 = Ra2 + 1
  ActiveSheet.Cells(Ra2, 1).Value = ArtNr
  ActiveSheet.Cells(Ra2, 2).Value = CDbl(Kvot)
  Workbooks(BudgetBok).Activate
End Sub
```

Om användaren matar i 1,1 som kontrollvärde visas följande rapport:

	A	B
1	Art nr	Kvot
2	45-350	1,226667
3	46-550	1,182353
4	55-350	1,12
5	65-350	1,101667
6	89-500	1,169565
7	96-550	1,104167
8	96-650	1,185714
9	96-750	1,135385

7.4 Do While...Loop

Under construction…

7.5 Select Case

Med Select Case kan olika sekvenser av kommandon utföras beroende på en numerisk variabels värde.

```
Dim intKod As Integer

Sub VäljaKod()
intKod = ActiveSheet.Cells(10, 2)    'Tilldelar variabel värde från cell B10
Select Case intKod                   'Select Case initieras
Case 1                               'Om värdet på IntKod är 1
  MsgBox "1"
Case 2, 4                            'Om värdet på IntKod är 2 eller 4
  MsgBox "2 & 4"
Case 3                               ' Om värdet på IntKod är 1
  MsgBox "3"
Case 6 To 10                         ' Om värdet på IntKod är mellan 6 o 10
  MsgBox "6 to 10"
Case Else
  MsgBox "resten"
End Select
End Sub
```

7.6 With End With

Med *With … End With* kan flera åtgärder utföras på ett förenklat sätt.
I exemplet nedan matas forst en formel in i cellerna A1 till A10 sedan formateras dessa celler utan att koden behover innehalla objektreferensen ActiveSheet…

```
Sub WithExempel()
  ActiveSheet.Cells(1, 1) = "Datum"
  With ActiveSheet.Range(Cells(2, 1), Cells(4, 1))
    .Value = "=TODAY()"
    .NumberFormat = "yymmdd"
  End With
End Sub
```

I exempelt matas även en formel in i cellan.

8 Egendefinierade funktioner

Med hjälp av funktionsmakron kan man förenkla vanligt förekommande beräkningar, som man har. Detta åstadkommes genom att man skapar egna funktionsmakron, vilka läggs till listan av övriga Excelfunktioner. Dessa egna funktionsmakron kan man sedan utnyttja när helst man önskar. Om man lagrar funktionsmakron i arbetsboken **Egna.xls** under den katalog där Excel är lagrat kommer det automatiskt att finnas till hands när man startar Excel.

8.1 Skapande av funktionsmakro

Ett funktionsmakro fungerar på samma sätt som en vanlig EXCELfunktion t ex SUMMA() eller MEDEL(). Man skapar ett eget funktions makro genom att genomföra följande steg:

1. Definiera formler och argument
2. Ta fram en ny Visual Basic-modul eller aktivera en befintlig
3. Skriv ordet ***Function*** följt av funktionens namn
4. Skriv det/de argument inom parentes åtskilda med komma (,)
5. - Tryck Enter (om raden är riktigt skriven blir ordet **funktion** blått)
6. Skriv programraderna, tryck Tab före och Enter efter varje rad
7. Skriv ***End function*** som sista rad

I följande exempel skapas en egendefinierad funktion för omvandling av Celsiusgrader till Fahrenheitgrader.

Definition av formler och argument
b=(9/5)*a+32 är en formel där b=resultatet i Fahrenheit och a=argument i form av Celsiusgrader

Ta fram ny modul
- Välj **Verktyg/Makro/Visual Basic Editor**
- Välj **Infoga/Modul**

Inmatning av funktionen
Det värde som funktionen skall returnera måste tilldelas funktionsnamnet före slutet på funktionen

```
Function Fahrenheit(Celsiusgrader)
  Fahrenheit = 9 / 5 * Celsiusgrader + 32
```

```
End Function
```

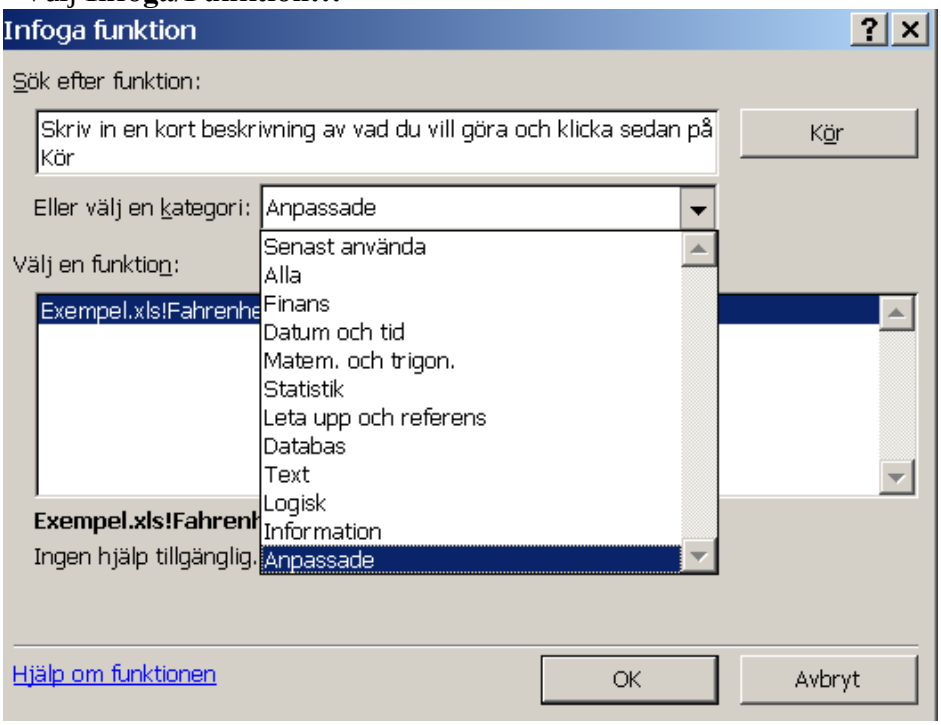
- Spara arbetsboken

Spara i XLStart
Om funktionsmakrot alltid skall vara tillgängligt, spara arbetsboken i mappen XLSTART, så kommer det att öppnas varje gång Excel startas. Mappen XLSTART finns normalt på följande sökväg: C:\Documents and Settings*användarnamn*\Application Data\Microsoft\Excel

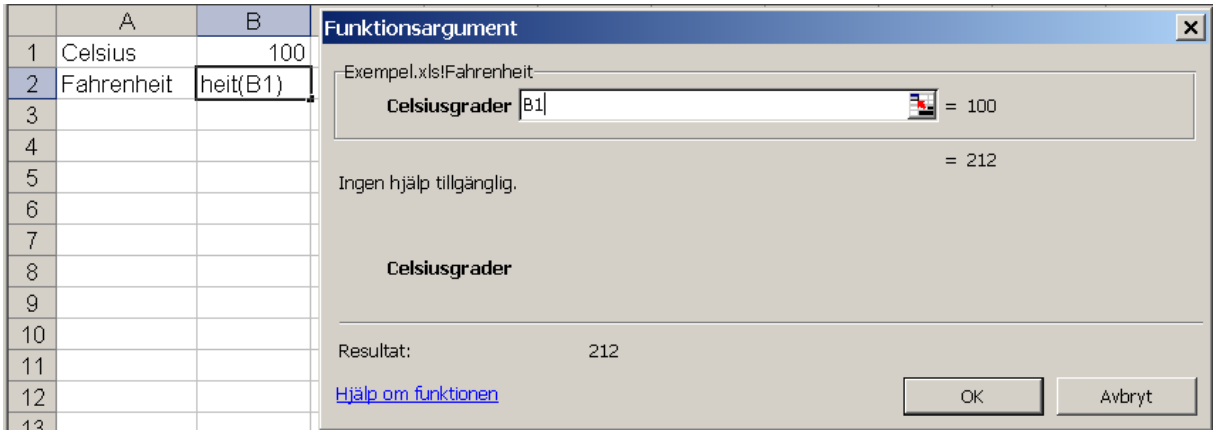
8.2 Användning av funktionsmakro

Användningen av funktionsmakro kräver att arbetsboken som funktionsmakrot är lagrat på är öppen.

För att beräkna Fahrenhietvärde med hjälp av funktionsmakrot:
- Ta fram en ny arbetsbok
- Skriv Celsius i A1 och Fahrenhet i A2
- Skriv 100 i B1
- Välj **Infoga/Funktion…**

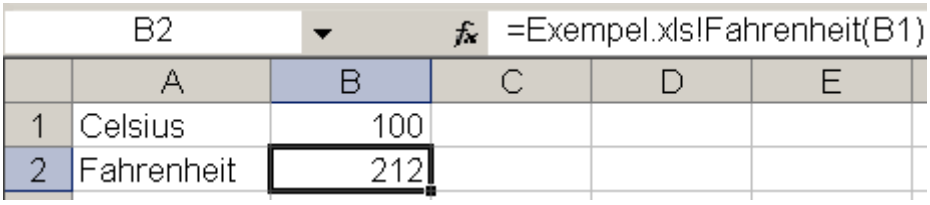


- Välj kategori **Anpassade**
- Välj Funktionsnamnet Fahrenheit
- OK



- Mata in B1 som argument
- OK

Så här blir det:



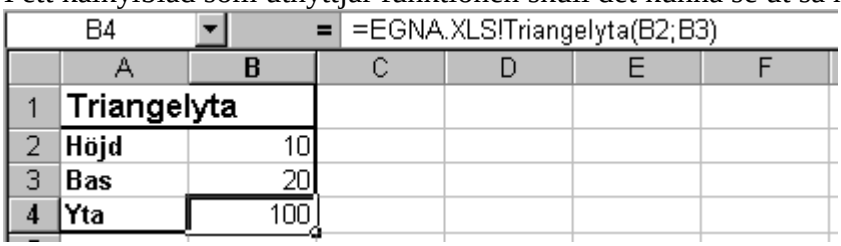
8.3 Exempel:

8.3.1 Triangelyta

Följande funktionsmakro beräknar ytan på en triangel. Funktionen har två argument: Höjd och Bas.

```
Function Triangelyta(Höjd, Bas)
  Triangelyta = Höjd * Bas / 2
End Function
```

I ett kalkylblad som utnyttjar funktionen skull det kunna se ut så här:



8.3.2 Exempel konkatenering

```
Function Konk(a as string, b as string) as string
  Konk = a & b    ' Concatenate a and b
End Function
```

8.3.3 Övningar: Funktionsmakro:

Gör egendefinierade funktionsmakron för:

Cirkelyta: Pi*radien * radien (3,14*radie^2)

Cirkelomkrets: Pi*radien*2

Rektangelomkrets (kräver två argument)

9 Dialoger

Att utforma en dialog i ett makro innebär att man ger möjlighet till användaren att mata in värden och programmet visar resultat av eventuella bearbetningar. Dialoger i VBA kan vara mer eller mindre komplicerade. Då man använder fördefinierade dialogrutor för inmatning är det enklare att skriva makrot. Om kraven på dialogen är lite mer omfattande får man utforma egna dialogrutor.

Fördefinierade dialoger använder sig av standard-dialogrutorna *Inputbox* och *MsgBox*.

9.1 Ett enkelt exempel

I nedanstående exempel beräknas rymden av ett klot. Detta sker genom att ett radievärde matas in med hjälp av funktionen *Inputbox*. När svaret beräknats visas det med hjälp av meddelanderutan *MsgBox*.

```
Sub Rymdberäkning1()
Dim Radie, Rymd
  Radie = InputBox("Ange radie")
  Rymd = (Radie ^ 3 * 4 * Application.Pi()) / 3
  MsgBox "Rymden blir: " & Rymd
End Sub
```

9.2 InputBox

Med funktionen *InputBox* kan användaren mata in värden till programmet under körningen. Inputbox har följande format:

InputBox(*ledtext* [,*rubrik*] [,*standard*] [, *xpos*] [, *ypos*] [,*hjälpfil*, *sammanhang*])

InputBox(*ledtext*[,*rubrik*][,*knappar*][,*standard*][,*hjälpfil*,*sammanhang*])

ledtext	Stränguttryck som visas i dialogrutan
rubrik	Stränguttryck som visas på dialorutans titelrad
standard	Stränguttryck som visas i dialogrutans textruta
xPos	Talvärde som bestämmer dialorutans position i x-led
yPos	Talvärde som bestämmer dialorutans position i y-led
hjälpfil	Anger hjälpfil
sammanhang	Talvärde som anger hjälpavsnitt i hjälpfil

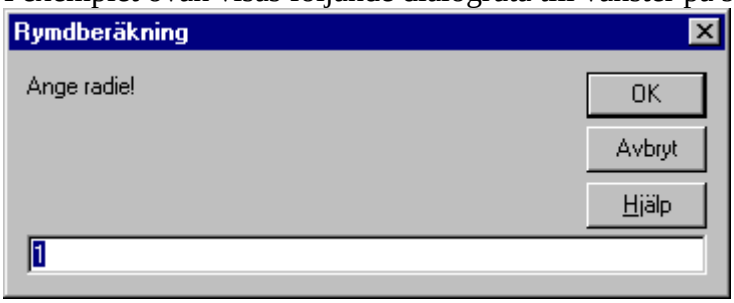
Dialogrutan centreras automatiskt på skärmen om argumenten för x- och y-position utelämnas. *InputBox* returnerar värdet som användaren har angett om OK valts eller RETUR. tryckts. En sträng med längden noll returneras om användaren väljer Avbryt.

9.2.1 Exempel 1 InputBox:

```
Sub Rymdberäkning2()
Dim Radie, Rymd
Dim Standard, xPos As Integer
Dim Ledtext, Sammanhang, Rubrik, Hjälpfil As String
  Ledtext = "Ange radie!"
  Rubrik = "Rymdberäkning"
  Standard = 1
  xPos = 1
  Hjälpfil = "SUPPORT.HLP"
  Sammanhang = 10
  Radie = InputBox(Ledtext, Rubrik, _
    Standard, xPos, , Hjälpfil, Sammanhang)
```

```
  Rymd = (Radie ^ 3 * 4 * Application.Pi()) / 3
  MsgBox "Rymden blir: " & Rymd
End Sub
```

I exemplet ovan visas följande dialogruta till vänster på skärmen.

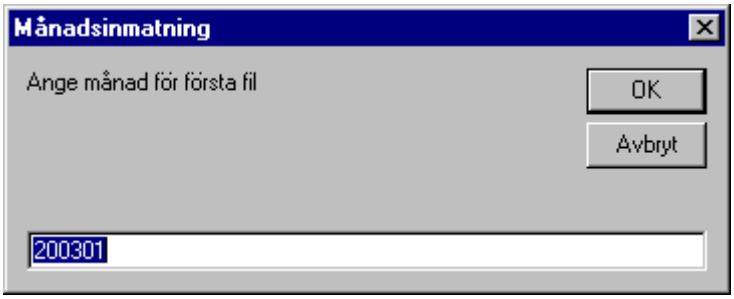


9.2.2 Exempel 2 InputBox:

I nedanstående exempel användsindataruta för att öppna en fil som har ett datum i namnet (t.ex. 200308MånProd.xls) där användaren får ett förslag på ett datum men kan ändra detta.

```
Konst Enhet = "g:\"
Konst Katalog = "exdata\lönsys\månad\"
Dim Mån1; Dbnamn; SökvägDb

Sub Filöppning()
  Dim Mån1, FilNamn, FilSökväg, Enhet, Katalog
  Enhet = "d:\"
  Katalog = "Exdata\"
  Mån1 = InputBox("Ange månad för första fil", "Månadsinmatning", "200301")
  FilNamn = Mån1 & "MånProd.xls"
  FilSökväg = Enhet & Katalog & FilNamn
  Workbooks.Open Filename:=FilSökväg, Password:="gurka"
End Sub
```



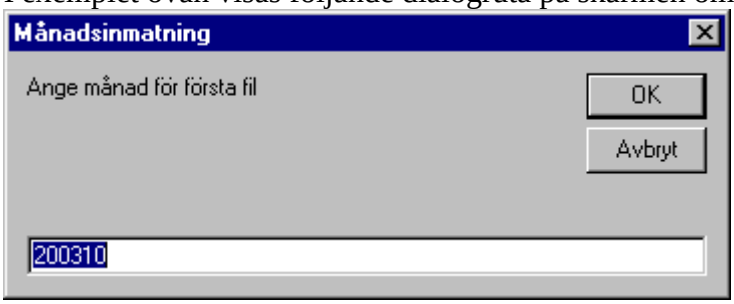
Detta alternativ strävar efter att uppnå ett visst mått av flexibilitet. Dels sparas enhets och katalognamn som konstanter vilket underlättar förändringar om de förekommer flera gånger. Dels ges användaren möjlighet att mata in valfri månad i en indataruta. I exemplet föreslås att filen 9612MÅN.XLS skall öppnas.

Ytterligare en utveckling kunde vara att låta programmet t.ex. föreslå månaden innan aktuell månad. Detta kunde utföras på följande sätt:

```
  Mån1 = InputBox("Ange månad för första fil", "Månadsinmatning", _
    Year(Now()) & Format(Month(Now()) - 1), "00")
```

- Funktionen **Now()** returnerar aktuellt datum och tid.
- Funktionen **Year(*tal*)** returnerar ett årtal med fyra siffror med **Now()** som argument.
- Funktionen **Month(*tal*)** returnerar ordningsnumret för den månad som datumuppgiften gäller. I exemplet gäller att om programmet körs i januari så returneras talet 1 och om det körs i december så returneras talet 12.
- I exemplet minskas månads numret med 1 för att få talet för föregående månad. Detta fungerar tyvärr inte i januari, men för övriga månader går det utmärkt. (Det går att lösa problemet med januari.)
- Funktionen **Format(*text, format*)** formaterar argumentet (*månadsnumret*) till formatet ”00” (eller något annat som anges) . Detta görs då dokumenten är sparade som 200301MånProd.xls, 200302MånProd.xls osv.

I exemplet ovan visas följande dialogruta på skärmen om rutinen körs i november 2003:



9.3 Avbryt

I nedanstående exempel hanteras möjligheten att användaren väljer *Avbryt*. Värdet som Inputbox returnerar sparas i variabeln **Radie**. Om användaren väljer *Avbryt* ges ett meddelande om att programmet avslutas varefter det avslutas med *Avsluta Sub*.

```
Sub Rymdberäkning3()
Dim Radie, Rymd
  Radie = InputBox("Ange radie", , 1)
  If Radie = "" Then
    MsgBox "Programmet avslutas!"
    End
  Else
    Rymd = (Radie ^ 3 * 4 * Application.Pi()) / 3
    MsgBox "Rymden blir: " & Rymd
  End If
End Sub
```

Ett annat exempel:

```
Dim startVecka ’ OBS deklareara ej som Integer
startVecka = InputBox("Ange startvecka för utskrift, (t.ex. 14)", , 1)
If startVecka = "" Then
  End
End If
```

9.4 MsgBox

Funktionen *MsgBox* används för att ge meddelanden till användaren.
MsgBox har följande format:
MsgBox(*ledtext*[;*knappar*][;*rubrik*][;*hjälpfil*;*sammanhang*])

ledtext	Stränguttryck som visas i dialogrutan
knappar	Talvärde som visar vilka knappar som ska visas i dialogrutan
rubrik	Stränguttryck som visas på dialorutans titelrad

hjälpfil Anger hjälpfil
sammanhang Talvärde som anger sammanhang

Konstant	Värde	Beskrivning
vbOKOnly	0	Visar endast OK
vbOKCancel	1	Visar OK och Avbryt
vbAbortRetryIgnore	2	Visar Avbryt, Försök igen och Ignorera
vbYesNoCancel	3	Visar Ja, Nej och Avbryt
vbYesNo	4	Visar Ja och Nej
vbRetryCancel	5	Visar Försök igen och Avbryt
vbCritical	16	Visar ”Kritiskt meddelande”-ikon
vbQuestion	32	Visar ”Frågetecken”-ikon
vbExclamation	48	Visar ”Varningstecken”-ikon
vbInformation	64	Visar ”Informationstecken”-ikon

Talvärden för returnerat svar:

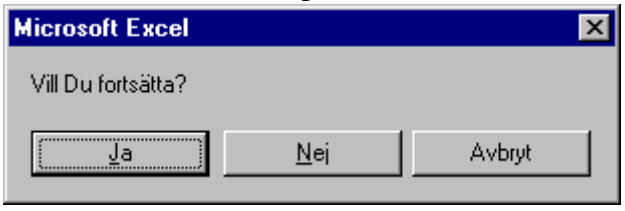
Konstant	Värde	Beskrivning
vbOK	1	OK
vbCancel	2	Avbryt
vbAbort	3	Avbryt
vbRetry	4	Försök igen
vbIgnore	5	Ignorera
vbYes	6	Ja
vbNo	7	Nej

9.4.1 Exempel 1 MsgBox

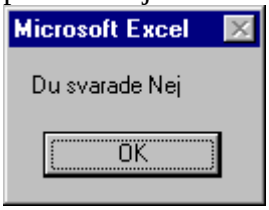
I detta exempel utförs olika saker beroende på vilken knapp användaren väljer.

```
Sub MsgBoxExempel1()
Dim Svar
  Svar = MsgBox("Vill Du fortsätta?", 3)
  If Svar = vbYes Then
    MsgBox ("Du svarade Ja")
  ElseIf Svar = vbNo Then
    MsgBox ("Du svarade Nej")
  End If
End Sub
```

Efter raden **`Svar = MsgBox("Vill Du fortsätta", 3)`** visas:



Om användaren väljer Ja respektive Nej:

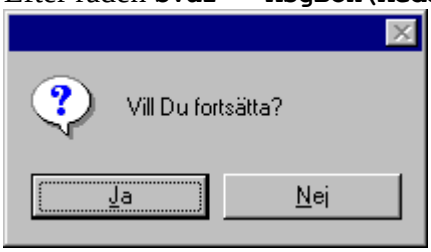


9.4.2 Exempel 2 MsgBox

I detta exempel kombineras konstanter.

```
Sub MsgBoxexempel2()
Dim Medd
Dim Knappar, Svar
  Medd = "Vill Du fortsätta?"
  Knappar = vbYesNo + vbQuestion
  Svar = MsgBox(Medd, Knappar)
  If Svar = vbYes Then
    MsgBox "Du svarade Ja!"
  Else
    MsgBox "Du svarade nej!"
  End If
End Sub
```

Efter raden **`Svar = MsgBox(Medd, Knappar)`** visas:



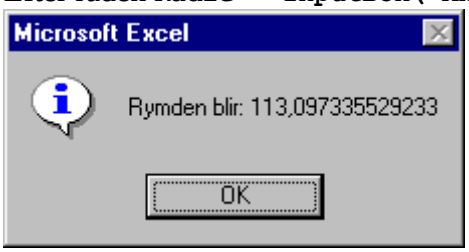
9.4.3 Exempel 3 MsgBox

Följande exempel visar svaret med en OK-knapp och ett informationsmärke:

```
Dim Radie, Rymd
Dim Medd
Dim Knappar, Svar
Sub Rymdberäkning()
  Radie = InputBox("Ange radie")
  Rymd = (Radie ^ 3 * 4 * Application.Pi()) / 3
  Medd = "Rymden blir: " & Rymd
  Knappar = vbOKOnly + vbInformation
  Svar = MsgBox(Medd, Knappar)
End Sub
```

Efter raden **`Svar = MsgBox(Medd, Knappar)`** visas:

Efter raden `Radie = InputBox("Ange radie")` visas:



10 Felhantering

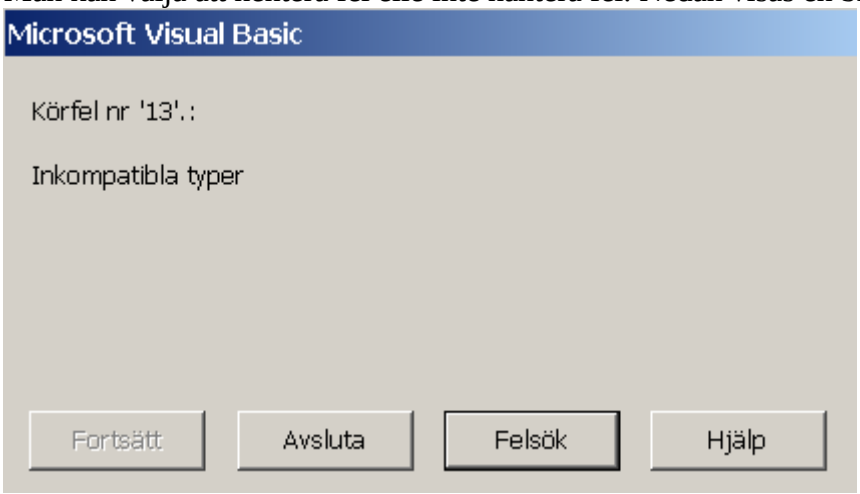
10.1 Olika feltyper

I VBA kan man indela fel i fyra olika typer, nämligen:

- Syntaxfel. Upptäcks då koden skrivs in på modulbladet.
- Kompileringsfel. Upptäcks då programmet undersöks innan det körs. En vanlig typ av kompileringsfel är odeklarerade variabler.
- Programkörningsfel. Uppstår då programmet körs. Kan t.ex. orsakas av falaktigt inmatade data.
- Logiska fel. Logiska fel kan inte upptäckas av programmet utan är en följd av felaktig programmering, som inte reulterar i något av ovanstående fel. Det kan t.ex. innebära att en körning ger fel resultat.

Felhantering ä viktigt i applikatione som skall användas i VBA finns möjligheter att ta hand om fel som uppträder vid programkörning.

Man kan välja att hentera fel elle inte hantera fel. Nedan visas en bild på ett icke hanterat fel.



För att hitta de olika felkodena sök i Hjälpen på Trappable Errors.

10.2 Ett enkelt exempel

Med hjälp av inbyggda funktione kan man hantera fel som uppstår. Nedan visas ett enkelt exempel på hu det kan gå till.

```
Sub TestaFelrutin()
Dim Svar As Integer

On Error GoTo ErrorHandler

  Svar = InputBox("Vad heter Du?")
  MsgBox ("Hej " & Svar)

  Exit Sub
ErrorHandler:
  MsgBox ("Fel fel")
End Sub
```

10.3 On Error Resume Next

I nedanstående exempel visas ett annat exempel på användning av programsatsen `On Error`. I exemplet skall användaren mata in ett kundnummer. Programmet söker reda på rätt rad i ett kundregister. Om användaren matar in ett kundnummer som inte finns, hanteras detta i koden. Med programsatsen `On Error Resume Next` aktiveras felhantering. Innebörden är att om ett fel uppstår, så fortsätter körningen på raden efter felet.
I exemplet kan raden `Workbooks(PrognosFil).Activate` resultera i fel om ett icke registrerat kundnummer matas in. På detta sätt undviks felet.

```
Sub Auto_Open()
  fileNamePath = ActiveWorkbook.Path
  On Error Resume Next
  Workbooks(PrognosFil).Activate
  If ActiveWorkbook.Name <> PrognosFil Then
    Workbooks.Open Filename:=fileNamePath & "\" & PrognosFil,
UpdateLinks:=3
  End If
  On Error Resume Next
  Workbooks(StycknFil).Activate
  If ActiveWorkbook.Name <> StycknFil Then
    Workbooks.Open Filename:=fileNamePath & "\" & StycknFil, UpdateLinks:=3
  End If
  GåTillMenyBoken '*** RUTIN ***
End Sub
```

11 Interaktion med användaren

11.1 Skärmuppdatering

Om man vill slippa en blinkande och flipprande bild då ett makro körs kan man stänga av med egenskapen *ScreenUpdating*. Denna är boleansk och kan alltså ges värdena *True* eller *False*.

```
  Application.ScreenUpdating = False

If i = 2 Then Application.ScreenUpdating = False
```

För att återställa skärmuppdatering i programmet:

```
  Application.ScreenUpdating = True
```

Skärmuppdateringen slå på sig själv när programmet avslutats.

11.2 Använda statusraden

Om man slår av skärmuppdateringen kan det vara lämpligt att upplysa användaren om vad som händer i programmet genom att visa meddelanden på *Statusraden*. Då kan egenskapen *StatusBar* användas.

```
For i = 1 To 10000
  Application.StatusBar = i
Next i
```

Genom att sätta egenskapen *DisplayStatusBar* (boolean) till *True* kan statusraden manipuleras. Genom att sätta den till *False* ges kontrollen tillbaka till Excel.

```
gammalStatusRad = Application.DisplayStatusBar
Application.DisplayStatusBar = True
Application.StatusBar = "Var god vänta..."
Workbooks.Open filename:="LARGE.XLS"
Application.StatusBar = False
Application.DisplayStatusBar = gammalStatusRad
```

På den första raden i exemplet sparas den text som fanns på statusraden. På den sista raden visas denna text igen på statusraden.

```
Arbetsbok = Cells(pRad1, 1) & ".xls"
FilKatalog = Cells(pRad1, 8)
Sökväg = FilKatalog & Arbetsbok
Workbooks.Open FileName:=Sökväg, UpdateLinks:=0, ReadOnly:=True
Application.StatusBar = "Nu behandlas: " & Sökväg
```

11.3 MsgBox

Medelanden kan ges via en dialogruta i progarammet. Funktionen för detta hetar *MsgBox*.

```
MsgBox ("Värdet är " & i)
```

11.4 Inmatning av värden

Då programmet behöver värden inmatade under körning av programmet kan funktionen *InputBox* användas. (Med *Default* i exemplet nedan avses ett standardvärde som används om inte användaren själv anger ett värde.

```
Dim Meddelande, Titel, Default, NyttVärde
Meddelande = "Ange ett värde mellan 1 och 4"    ' Ange ledtext.
Titel = "InputBox Demo"    ' Ange Titel.
Default = "1"    ' Ange default.
' Visa Meddelande, Titel, and defaultvärde.
NyttVärde = InputBox(Meddelande, Titel, Default)

' Använd hjälpfil och context. Hjälpknappen läggs till automatiskt.
NyttVärde = InputBox(Meddelande, Titel, , , , "DEMO.HLP", 10)

' Visa dialog box vid position 100, 100.
NyttVärde = InputBox(Meddelande, Titel, Default, 100, 100)
```

12 Diverse

12.1 Använda Excelfunktioner i makro

I programmet kan Excelfunktioner användas tillsammans med objektet *Application*. Funktionerna skrivs på engelska.

Exemplen nedan utgår från följande kalkylblad:

	A	B	C	D	E	F
1	3	9	6	92	54	8
2	30	62	82	12	75	5
3	29	26	58	64	25	53
4	94	63	37	3	34	8
5	9	49	78	13	63	77
6	96	44	45	17	34	24
7	92	90	10	90	31	10
8	2	62	53	33	28	85
9	13	86	32	92	44	74
10	32	74	97	82	99	85
11	94	15	25	2	69	81
12	21	59	92	20	31	78

```
Dim Område As Range
Dim MinstaVärde
Sub HämtaMinstaVärde()
  Set Område = Worksheets("Blad1").Range("A1:F12")
  MinstaVärde = Application.WorksheetFunction.Min(Område)
  MsgBox MinstaVärde
End Sub
```

När Excelfunktionerna utnyttjas måste deras engelska benämningar användas i programmet. i nedanstående exempel används funktionen PASSA vars engelska benämning är MATCH.

```
Sub HittaFörsta()
Dim Plats
  Plats = Application.WorksheetFunction.
    Match(2, Worksheets(1).Range("A1:A12"), 0)
  MsgBox Plats
End Sub
```

I exemplet visas värdet

12.2 Öka hastigheten på programmet

Nedanstående åtgärder ökar hastigheten på program.

- Avstå från **select** i största möjliga utsträckning
- **Application.screenupdating = false** ' slå av skärmekot i början av programmet
- Application.screenupdating = true 'kan läggas in i slutet, men i allmänhet kommer det tillbaka ändå
- Sätt omräkning till manuell (**Application.Calculation = xlCalculationManual**), men se till att det inte blir fel värden som kommer ut från programmet. Sätt tillbaka till automatisk innan några omräknade värden används.

12.3 Systemparametrar

I VBA finns mänder med parametrar med vars hjälp man kan få systeminformation av olika slag.

I nedanstående exempel

```
Sub Bank_File()
Dim BankFile As String
  BankFile = Application.GetOpenFilename
  ´Kod
End Sub
```

12.4 Att aktivera, referera och markera

12.4.1 Välja filer

I nedanstående exempel kommer metoden GetOpenFilename att visa dialogboxen Öppna. Därefter kan användaren välja en fil som finns tillgänglig via variabeln BankFile.

```
Option Explicit
Sub Bank_File()
Dim BankFile As String
  MsgBox "Välj en textfil"
  BankFile = Application.GetOpenFilename
  Workbooks.OpenText Filename:=BankFile
End Sub
```

12.4.2 Öppna filer

Workbooks.Open Enh & Kat1 & KatÅr & Kat2 & Filnamn 'Ändring 981220

Nedan presenteras några alternativa sätt att öppna en fil. Dessa kommandon är även lätta att spela in.

Alt 1:

```
Workbooks.Open "G:\EXDATA\LÖNSYS\ACK1\200201ACK.XLS"
```

Alt 2:

```
ChDir "G:\EXDATA\LÖNSYS\ACK1"
Workbooks.Open "200201ACK.XLS"
```

I detta alternativ görs först katalogen där filen finns. Detta innebär att filer sparade på denna katalog kan behandlas utan att sökväg anges.

Alt 3:

```
Const Enhet = "g:\"
Const Katalog = "exdata\lönsys\månad\"
Dim Mån1; Dbnamn; SökvägDb

Sub Filöppning()
  Mån1 = Indataruta("Ange månad för första fil"; "Månadsinmatning"; "0312")
  Dbnamn = Mån1 & "MÅN.xls"
  SökvägDb = Enhet & Katalog & Dbnamn
  Workbooks.Open Filename:=SökvägDb; Password:="gurka"
Slut Sub
```

Detta alternativ är lite mer flexibelt än de två första. Dels sparas enhets och katalognamn som konstanter vilket underlättar förändringar om det förekommer flera gånger. Dels ges användaren möjlighet att mata in valfri månad i en indataruta. I exemplet föreslås att filen

0312MÅN.XLS skall öppnas. Se avsnittet om *Indataruta* för ytterligare kommentarer och utvecklingsförslag.

Från hjälpen:
```
Workbooks.Open "ANALYSIS.XLS"
ActiveWorkbook.RunAutoMacros xlAutoOpen
```
Gör det möjligt att köra automakron

12.4.3 Stänga filer

För att stänga en arbetsbok:
```
Workbooks("Budget2004.xls").Close SaveChanges:=True
```
Om ändringar ej skall sparas sätts sista värdet till `False` istället för `True`.
Om en variabel används som dokumentnamn, när t.ex. flera dokument öppnas och stängs av programmet, visas av följande exempel:
```
Workbooks(BudgetDok).Close SaveChanges:=True
```

Följande exempel visar hur alla öppna arbetsböcker stängs.
```
Workbooks.Close
```
Om ändringar gjorts kommer Excel, via dialogrutor, att ge användaren möjlighet att spara.

12.4.4 Skapa filer

Nedanstående exempel visa hur en fil skapas och används i ett makro.
```
  Workbooks.Add 'Ny bok för rapport
  rptNamn = ActiveWorkbook.Name
  Workbooks(rptNamn).Worksheets("Blad1").Cells(1, 1) = _
"Faktureringsbevakning " & numFromMonth & " " & numFromYear
```

12.4.5 Kontrollera att filer finns

För att kontrollera att viss fil finns kan funktionen *Dir* användas.

Nedanstående exempel kontrollerar om en fil finns, öppnar den om den finns och meddelar användaren om den saknas.
```
  SökVägFilNamn = "Budget2004.xls"
  Filer = ""
  Filer = Dir(SökVägFilNamn)
  If Filer <> "" Then
    Workbooks.Open Enh & Kat1 & KatÅr & Kat2 & Filnamn
  Else
    MsgBox "Filen saknas!"
  End If
```
Observera att variabeln Filer måste nollställas (`Filer = ""`) om man skall öppna fler filer efter varandra.

I nedanstående exempel ersätter en ny fil en befintlig utan att användaren tillfrågas.
Funktionen *SendKeys* fungerar som en tangenttryckning.
```
  SökVägFilNamn = "D:\Data\Budget2003.xls"
  Filer = ""
  Filer = Dir(SökVägFilNamn)
  If Filer <> "" Then
    Application.SendKeys ("j")
  End If
  Workbooks.Open SökVägFilNamn
```

Funktionen SendKeys fungerar som en tangenttryckning.

Från Hjälpen:
' Display the names in C:\ that represent directories.
```
MyPath = "c:\"    ' Set the path.
MyName = Dir(MyPath, vbDirectory)    ' Retrieve the first entry.
Do While MyName <> ""    ' Start the loop.
    ' Ignore the current directory and the encompassing directory.
    If MyName <> "." And MyName <> ".." Then
        ' Use bitwise comparison to make sure MyName is a directory.
        If (GetAttr(MyPath & MyName) And vbDirectory) = vbDirectory Then
            Debug.Print MyName    ' Display entry only if it
        End If    ' it represents a directory.
    End If
    MyName = Dir    ' Get next entry.
Loop
```

OBS: Näst sista raden visar hur nästa fil hämtas till behandling

12.4.6 Spara filer

Om man vill stänga en fil och spara den samtidigt:
```
  Workbooks("Orderlista.xls").Close saveChanges:=True
```
eller
```
  ActiveWorkbook.Close SaveChanges:=True
```
Om man vill spara en fil utan att stänga den:
```
  Workbooks(Filnamn).Close saveChanges:=False
```

Då man vill spara en ny fil kan man göra på följande sätt:
Alt 1:
```
ActiveWorkbook.SaveAs Filename:="D:\Uppföljning\Total\Rapport.xls"
```

Alt 2 (med variabler):
```
SökvägDatabas = Enhet & KatalogDb & Dbnamn
  ActiveWorkbook.SaveAs Filename:=SökvägDatabas
```

Alt 3 (med lösenord):
```
SökvägDatabas = Enhet & KatalogDb & Dbnamn
  ActiveWorkbook.SaveAs Filename:=SökvägDatabas; Fileformat:=xlNormal; _
    Password:="gurka"
```

Om en fil redan finns med samma namn som den som skall sparas och man vill ersätta den gamla filen automatiskt:
```
  RappNamn2 = Enh & "\" & RappPath & "\" & RappNamn1 & ".xls"
    FilerRapp = ""
    FilerRapp = Dir(RappNamn2)
    If FilerRapp <> "" Then 'Om rapporten redan finns
      Application.SendKeys ("j") 'Skall den ersättas
    End If
  ActiveWorkbook.SaveAs Filename:=RappNamn2
```

12.5 Att utföra diverse saker

12.5.1 Att använda Exceldatabasen och Urval i programmet

Databasen

	A	B	C	D	E	F
1	**Art nr**	**Artikelbenämning**	**Grundkod**	**Grkbenämning**	**Andel**	**Djurslag**
2	403708	Kotlett m ben Piggham	403708	Kotlett	0,38	Gris
3	403709	Kotlett m ben Annat	403708	Kotlett	0,45	Gris
4	**420300**	**Bog**	**420300**	**Bog**	**100%**	Konv nöt
5	420334	nötbog scan vac	420300	Bog	100%	Konv nöt
6	421905	Bogstek scan sk nöt	420300	Bog	50%	Konv nöt

Villkorsområde och Urvalsområde

	H	I	J	K	L	M	N	O	P
	Art nr	**Artikelbenämning**	**Grundkod**	**Grkbenämning**	**Andel**	**Djurslag**	**Urval**	**Grundkod**	**Grkbenämning**
								403708	Kotlett
	Villkor							**420300**	**Bog**
								421200	**Högrev**
								422200	**Ryggbiff**
								423246	**Ryggbiff II**

Programmet nedan utnyttjar de båda bladen för att göra ett uval och lägga i Urvalsområdet:

```
Sub VäljGrundkoder()
  Set ObjArtBlad = Workbooks(MenyBlad).Sheets("Artiklar")
  AktDjurslag = Cells(12, 6)
  Sheets("Artiklar").Visible = True
  Workbooks(MenyBlad).Sheets("Artiklar").Select
  ObjArtBlad.Cells(2, 13) = AktDjurslag
  Range("A1:F2000").AdvancedFilter Action:=xlFilterCopy,
CriteriaRange:=Range( _
    "H1:M2"), CopyToRange:=Range("O1:P1"), Unique:=True
  ActiveWindow.SelectedSheets.Visible = False
  Workbooks(MenyBlad).Sheets("Meny").Select
End Sub
```

12.5.2 Utskrift

Exempel:

```
Sub UtskriftAllaGrundkoder()
  startVecka = InputBox("Ange startvecka för utskrift, (t.ex. 14)")
  VäljVecka '***   RUTIN   ***
  aRad = 2
  While Sheets("Grundkoder").Cells(aRad, 1) <> ""
    AktGrundkod = Format(Sheets("Grundkoder").Cells(aRad, 1), "@")
    Windows("Prognos.xls").Activate
    Sheets(AktGrundkod).Select
    UtskriftsRutinen '***   RUTIN   ***
    aRad = aRad + 1
    GåTillMenyBoken '***   RUTIN   ***
  Wend
End Sub

Sub VäljVecka()
  If startVecka > förstaVecka Then
    startVecka = startVecka - (förstaVecka - 2)
  Else
    startVecka = startVecka + förstaVecka - 44
  End If
End Sub

Sub UtskriftsRutinen()
  ActiveSheet.PageSetup.PrintArea = _
  Range(Cells(4, startVecka), Cells(51, startVecka + 12)).Address
  ActiveWindow.SelectedSheets.PrintPreview
  ActiveSheet.PrintOut
End Sub
```

12.5.3 Datum och tid

Användningav **Date** för aktuellt datum.

```
  Worksheets("Data").Cells(1, 19) = Date
```

Användning av **Now()** för aktuell tid

Se även sid 42

13 Mer om variabler

I detta avsnitt behandlas varibler något mer fördjupat än vad som ggjorts i avsnitt 4.1.2.

13.1 Matriser

Matriser används för att lagra flera variabler av samma typ.

```
Dim Radmatris(20)
Dim i As Integer
Sub Radmatris()
  For i = 1 To 12
    Radmatris(i) = i * 2
  Next i
  For i = 1 To 12
    Cells(i, 1) = (Radmatris(i))
  Next i
End Sub
```

13.2 Deklaration av variabler

Variabler kan deklareras som public eller private. En variabel som deklarerats som **Private** kan endast användas i den modul som den är deklarerad inom. En **Public** variabel ä tillgänglig i alla procedurer i alla moduler.

13.3 Ändring av variablers datatyp

Vissa variablers datatyp kan (under vissa omständigheter) ändras andras kan det inte.

```
CBool(expression)
CByte(expression)
CCur(expression)
CDate(expression)
CDbl(expression)
CDec(expression)
CInt(expression)
CLng(expression)
CSng(expression)
CStr(expression)
CVar(expression)
```

```
Dim startVecka
Dim utskrVecka1 As Integer
Const förstaVecka = 49

  startVecka = InputBox("Ange startvecka för utskrift, (t.ex. 14)")
  If CInt(startVecka) > förstaVecka Then
    utskrVecka1 = CInt(startVecka) - (förstaVecka - 2)
  End If
```

```
Dim startVecka As Integer
Const förstaVecka = 49

  startVecka = InputBox("Ange startvecka för utskrift, (t.ex. 14)")
  If startVecka > förstaVecka Then
    utskrVecka1 = startVecka - förstaVecka - 2
  End If
```

Följande kod ändrar ett numeriskt värde till text

```
      pGrundKod = Format(Sheets("Grundkoder").Cells(pRad, 1), "@")
```

14 Index