

## Övningar

1. Du har tidigare gjort en extern applikation i Visual Basic.  
Skapa ett gränssnitt varifrån denna kan startas.
2. Skapa ett Visual Basic gränssnitt med en bildruta som har en DDE länk till någon annan applikation du har tillgång till.  
Om du inte har någon sådan applikation skriv ändå programmet som om du hade det.

## 16 Ett större exempel

### Att deklarera argument som kontrollobjekt

Som avslutning ska vi göra ett större program som innehåller några av de moment vi gått igenom.

Innan vi börjar ska vi hålla en liten överkurs. Vi kommer här att deklarera en del argument som *kontroller*. Exempelvis

#### Procedure LäggUt (Kort As Control)

Här har vi deklarerat *Kort* som en kontroll. Vi antar att vi har textrutorna *Kort1*, *Kort 2* och *Kort3*, och att vi för alla tre vill utföra samma långa och krångliga beräkning. Någonstans i mitten av beräkningen ska vi skriva ut "spader" i respektive textruta. Om vi inte hade möjligheten att deklarera argument som kontroller skulle vi då vara tvungna att skriva tre nästan likadana procedurer - det enda som skulle skilja dem åt skulle vara raden där "spader" skrevs ut i en av textrutorna.

Genom att deklarera *Kort* som kontroll räcker det med en procedur. Vi skriver på följande sätt:

#### Procedure LäggUt (Kort As Control)

Lång  
krånglig  
beräkning

## KAPITEL 16

Kort.Text = "spader"

*fortsättning  
på  
lång  
krånglig  
beräkning*

Raden *Kort.Text = spader* gör att spader skrivs ut i textrutan för Kort1, Kort2 eller Kort3, beroende på vilket kort som givits som argument till LäggUt. Om vi anropar LäggUt med argumentet Kort2, dvs vi skriver

LäggUt Kort2

så kommer *Kort* att få värdet *Kort2*, och raden

Kort.Text = "Spader"

kommer internt att göras om till

Kort2.Text = "Spader"

## En pokermaskin

Låt oss nu börja med exemplet. Vi ska göra en sorts pokermaskin som man kan spela med antingen ensam eller upp till fyra deltagare.

Reglerna är följande: Till att börja med visas fem kort upp. Användaren får klicka på de kort som ska bytas. Man får sedan poäng efter de händer som kommer upp enligt nedan:

*par*  
(två kort av samma valör, tex två treor)

*två par*  
(t ex två treor och två damer)

*triss*  
(tre kort av samma valör t ex tre åttor)

*stege*  
(fem kort i sekvens, t ex 5-6-7-8-9)

*färg*  
(fem kort i samma färg, t ex fem hjärter)

*kåk*  
(en triss och ett par, t ex tre sjuor och två fyror)

*fyrtal*  
(fyra av samma valör, t ex fyra knektar)

*straight flush*  
(en stege i färg, t ex hjärter sex, till och med hjärter tio)

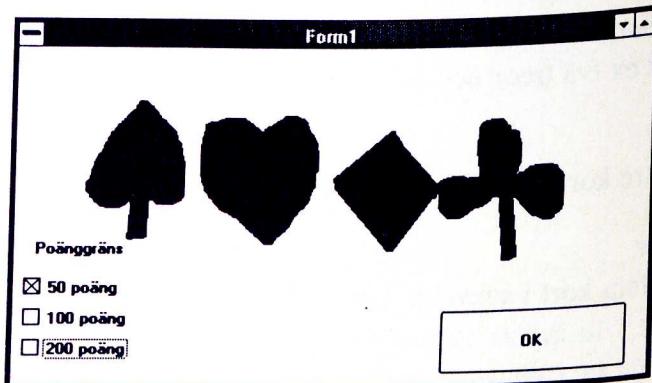
*femtal*  
(fem av samma valör, t.ex fem kungar)

Deltagarna får alltså poäng efter de händer de har, och kan sedan tävla om att komma först till 50, 100 eller 200 poäng. För att göra det lite enklare för oss antar vi ett obegränsat antal kortlekars, dvs vi kan få hur många som helst av samma kort.

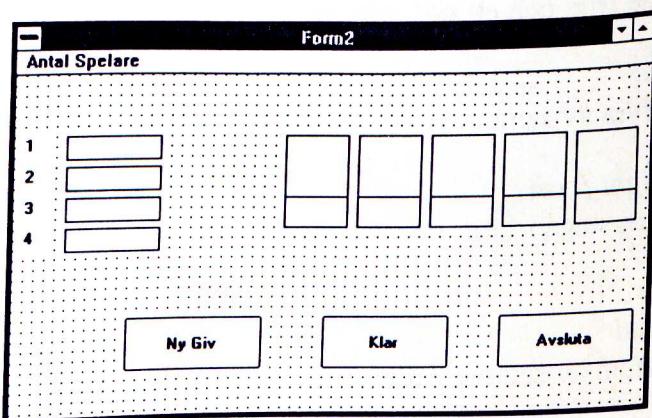
När applikationen körs igång kommer följande formulär upp där användaren får ange vilken poänggräns de vill tävla till.



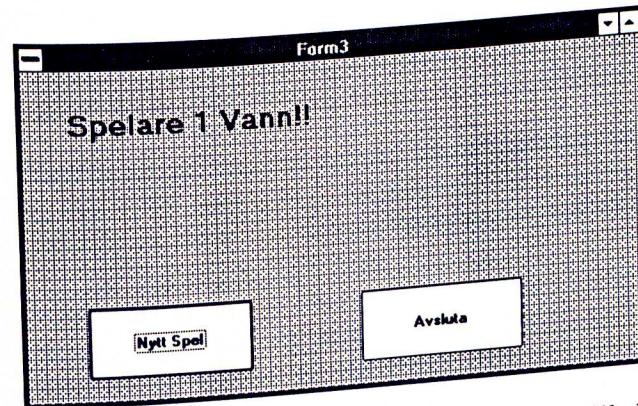
## KAPITEL 16



När de trycker på OK kommer det egentliga spelet upp:



När någon av spelarna vunnit kommer slutligen följande formulär upp:



Vi börjar med att skapa gränssnitten. Bakgrunden till det första formuläret kan vi rita upp i t ex PaintBrush. Efter att ha skapat lämpligt konstverk startar vi Visual Basic.

1. Välj egenskapen Picture i egenskapslistan.
2. Klicka på de tre knapparna i värdelistan och leta fram din fil från PaintBrush.
3. Lägg ut övriga kontrollobjekt.

Det sista formuläret kan vi också göra i PaintBrush eller, om man är lat, bara färglägga med Färgpaletten. Vi väljer att vara lata:

1. Gå in i menyn *Fönster* och välj *Färgpaletten*.
2. Klicka på lämplig färg.

Mittenfönstret består av ett stort antal kontrollobjekt som ska läggas ut. Kartan består av en bildruta (överst) och en textruta. Vi behöver därför:

Fem bildrutor till kartan.

Nio textrutor: fem till kartan och fyra att skriva poäng i.

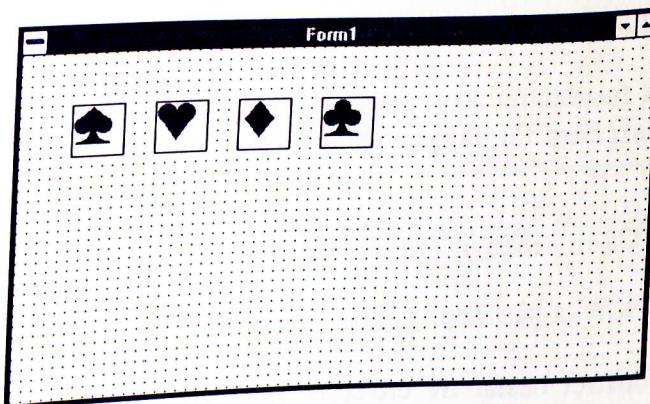
Fyra etiketter bredvid poängrutorna.

Tre kommandoknappar med texterna Ny Giv, Klar och Avsluta.

En meny med namnet "Antal Spelare" som innehåller alternativen "En spelare", "Två spelare", "Tre spelare" och "Fyra spelare".

Vi kommer här att anta kodnamnen *Kort1*, *Kort2*, *Kort3*, *Kort4* och *Kort5* på bildrutorna, *KortEtt*, *KortTvå*, *KortTre*, *KortFyra*, *KortFem*, *Spelare1*, *Spelare2*, *Spelare3* och *Spelare4* på textrutorna, *DeltMen* på menyn och *GivKnp*, *KlarKnp*, och *SlutKnp* på kommandoknapparna. Etiketterna får ha kvar de kodnamn de har som default, eftersom vi inte kommer att använda dem i programmet.

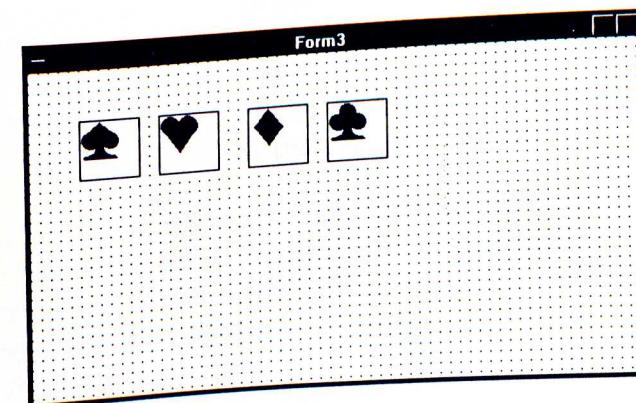
Efter att ha skapat gränssnittet fortsätter vi med korten. Vi använder oss av de fördefinierade ikonerna i IKONER/BLANDAT:



Vi ska alltså göra fyra bildrutor, en för respektive färg. Vi gör detta i ett formulär som vi kallar Kort:

1. Gå in i menyn Arkiv och välj Nytt Formulär.
2. Sätt egenskapen *FormName* till *Kortfönster*.

3. Lägg ut fyra bildrutor, helst av samma storlek som de i förra formuläret.
4. Ta fram egenskapen *Picture* för en av bildrutorna.
5. Klicka på de tre prickarna till höger om värdelistan för att få upp tillgängliga filer.
6. Klicka på *IKONER*, sedan på *BLANDAT*, sedan på *BLAND37*. Du har nu spadersymbolen i bildrutan.
7. Ge bildrutan kodnamnet *Spader*.
8. Ge de andra bildrutorna kodnamnen *Hjärter*, *Ruter* och *Klöver* och ladda in respektive ikon. Hjärter har filnamnet *BLAND34*, ruter filnamnet *BLAND36* och klöver filnamnet *BLAND35*.



Vi fortsätter med att deklarera alla globala variabler. Gå in i filen *Global.Bas* genom att dubbelklicka på den i projektfönstret och skriv

`Global maxpoäng As Integer`

`Global Spelare As String`

`Global Antal As Integer`

## KAPITEL 16

```

Global Spelare1 As String
Global Spelare2 As String
Global Spelare3 As String
Global Spelare4 As String

Global Färg1 As String
Global Färg2 As String
Global Färg3 As String
Global Färg4 As String
Global Färg5 As String

Global Värde1 As Integer
Global Värde2 As Integer
Global Värde3 As Integer
Global Värde4 As Integer
Global Värde5 As Integer

Global ByttKort1 As Integer
Global ByttKort2 As Integer
Global ByttKort3 As Integer
Global ByttKort4 As Integer
Global ByttKort5 As Integer

Global Spelare1poäng As Integer
Global Spelare2poäng As Integer
Global Spelare3poäng As Integer
Global Spelare4poäng As Integer

Global Const BLACK = &H0&
Global Const RED = &HFF&
Global Const GREEN = &HFF00&
Global Const YELLOW = &HFFF&
Global Const BLUE = &HFF0000
Global Const MAGENTA = &HFF00FF
Global Const CYAN = &HFFFF00
Global Const WHITE = &HFFFFFF

```

Vi fortsätter med koden för det första formuläret. Här ska användaren ange vilken poänggräns som ger vinst, och sedan klicka på *OK*. Om användaren inte angivit någon poänggräns kommer en Inputruta upp som ber användaren ange en sådan. Det tredje argumentet till *InputBox*, 50, är det värde InputBox får om användaren trots allt väljer att inte fylla i något. Poänggränsen blir då satt till 50.

```

Sub OK_Click ()
    If Check1.Value = 1 Then
        Maxpoäng = 50
    ElseIf Check2.Value = 1 Then Maxpoäng=100
    ElseIf Check3.Value=1 Then Maxpoäng=200
    Else Maxpoäng = Val(InputBox$("Ange den
        poäng du spelar till", "Pokermaskin",
        "50"))
    End If
    Spelare = "Spelare4"
    Load Form2
    Form2.Show
End Sub

```

Koden för sista formuläret, dvs det som talar om vem som vunnit, är också enkel. Vi har två kommandoknappar där den ena startar om spelet och den andra avslutar det. När vi startar om spelet måste vi nollställa alla spelarpoäng. Sedan laddar vi nästa formulär och visar det.

```

Sub Command1_Click ()

```

## KAPITEL 16

```

    Spelare1poäng = 0
    Spelare2poäng = 0
    Spelare3poäng = 0
    Spelare4poäng = 0
    Spelare5poäng = 0
    Load Form2
    Form2.Show

End Sub

Sub Command2_Click ()
    End
End Sub

```

Slutligen ska vi skriva koden för mittenfönstret där själva spelet försiggår.

När användaren klickar på *Ny Giv* ska fem nya kort komma upp. Det enda vi ska göra här är att anropa proceduren *NyGiv*. *NyGiv* anropar i sin tur proceduren *LäggUtKort* som vi definierar nedan.

Eftersom *NyGiv*, liksom ett antal andra procedurer i det här programmet, anropas från flera formulär skapar vi en *modul* där vi definierar proceduren. En modul innehåller procedurer och funktioner som kan anropas från alla formulär, på samma sätt som en global fil innehåller variabler och konstanter som kan anropas från alla formulär.

För att få en modul att skriva i går du in i menyn *Arkiv* och väljer *Ny Modul*. I modulen kommer vi sedan att skriva alla "egna" procedurer. Av de tre procedurerna nedan är alltså *Giv-Knp\_Click* skrivet i formuläret där knappen finns, medan *NyGiv* och *LäggUtKort* är skrivna i modulen.

```
Sub GivKnp_Click ()
```

```

NyGiv

End Sub

Sub NyGiv ()
    Dim Temp As String

    ByttKort1 = 0
    ByttKort2 = 0
    ByttKort3 = 0
    ByttKort4 = 0
    ByttKort5 = 0
    If Antal = 0 Then Antal = 1
    Select Case Antal
        Case 1
            Temp = "Spelare1"
        Case 2
            If Spelare = "Spelare1" Then
                Temp = "Spelare2"
            Else Temp = "Spelare1"
            End If
        Case 3
            If Spelare = "Spelare1" Then
                Temp = "Spelare2"
            ElseIf Spelare = "Spelare2" Then
                Temp = "Spelare3"
            Else Temp = "Spelare1"
            End If
    End Select
    LäggUtKort Temp

```

KAPITEL 16

```

Case 4
If Spelare = "Spelare1" Then
    Temp = "Spelare2"
ElseIf Spelare = "Spelare2" Then
    Temp = "Spelare3"
ElseIf Spelare = "Spelare3" Then
    Temp = "Spelare4"
Else Temp = "Spelare1"
End If
End Select
Spelare = Temp
LäggUtKort Form2.Kort1, Form2.KortEtt,
Färg1, Värde1
LäggUtKort Form2.Kort2, Form2.KortTvå,
Färg2, Värde2
LäggUtKort Form2.Kort3, Form2.KortTre,
Färg3, Värde3
LäggUtKort Form2.Kort4, Form2.KortFyra,
Färg4, Värde4
LäggUtKort Form2.Kort5, Form2.KortFem,
Färg5, Värde5
End Sub

```

234

```

Sub LäggUtKort (KortFrg As Control,
KortVal As Control, Färg$, Värde%)
Kort Färg, Värde
Select Case Färg
Case "Spader"
    KortFrg.Picture=Kortfönster.Spa
    der.Picture
Case "Hjärter"
    KortFrg.Picture=Kortfönster.Hjär
    ter.Picture
Case "Ruter"
    KortFrg.Picture=Kortfönster.Ru
    ter.Picture
Case "Klöver"
    KortFrg.Picture=Kortfönster.Klö
    ver.Picture
End Select
If Färg = "Hjärter" Or Färg = "Ruter" Then
    KortVal.ForeColor = RED
Else KortVal.ForeColor = BLACK
End If
KortVal.Text = Str$(Värde)
End Sub

```

Proceduren *Kort* använder slumptalsgeneratorn för att få fram ett tal mellan ett och femtiotvå. Vi låter talen mellan 1 och 13 vara spader ess till spader kung, talen mellan 14 - 26 vara hjärter ess till hjärter kung, etc. *Randomize* initierar generatorn och *Rnd* tar fram talet. Funktionen *Int* gör att vi får ett heltal.

```
Sub Kort (Färg As String, Värde As Integer)
```

235

## KAPITEL 16

```

Randomize
K = Int((52*Rnd + 1)
If K < 14 Then
    Färg = "Spader"
ElseIf K > 13 AND K < 27 Then Färg =
    "Hjärter"
ElseIf K > 26 AND K < 40 Then Färg =
    "Ruter"
ElseIf K > 39 Then Färg = "Klöver"
End If
Select Case K
Case 1, 14 , 27, 40
    Värde = 1
Case 2, 15 , 28, 41
    Värde = 2
Case 3, 16 , 29, 42
    Värde = 3
Case 4, 17 , 30, 43
    Värde = 4
Case 5, 18 , 31, 44
    Värde = 5
Case 6, 19 , 32, 45
    Värde = 6
Case 7, 20 , 33, 46
    Värde = 7
Case 8, 21 , 34, 47
    Värde = 8

```

```

Case 9, 22 , 35, 48
    Värde = 9
Case 10, 23 , 36, 49
    Värde = 10
Case 11, 24 , 37, 50
    Värde = 11
Case 12, 25 , 38, 51
    Värde = 12
Case 13, 26 , 39, 52
    Värde = 13
End Select

```

End Sub

Om användaren klickar på det första kortet ska det bytas. Vi börjar med att kontrollera att användaren inte redan bytt kortet, dvs om ByttKort1 har värdet Sant. I så fall kommer ett varningspip och inget annat händer.

Om kortet inte är bytt sätter vi ByttKort1 till Sant så att inga fler byten ska ske. Sedan rensar vi bildrutan och textrutan som representerar kortet och lägger ut ett nytt kort.

```

Sub Kort1_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)

```

```

If ByttKort1 = -1 Then
    Beep
Else ByttKort1 = -1
    Kort1.Cls
    KortEtt.Text = " "
    LäggUtKort Kort1,KortEtt,Färg1,Värde1
End If

```

End Sub

## KAPITEL 16

Vi gör likadant för övriga kort:

```

Sub Kort2_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)
If ByttKort2 = -1 Then
    Beep
Else ByttKort2 = -1
    Kort2.Cls
    KortTvå.Text = " "
    LäggUtKort Kort2,KortTvå,Färg2,Värde2
End If

End Sub

Sub Kort3_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)
If ByttKort3 = -1 Then
    Beep
Else ByttKort3 = -1
    Kort3.Cls
    KortTre.Text = " "
    LäggUtKort Kort3,KortTre,Färg3,Värde3
End If

End Sub

Sub Kort4_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)

```

```

If ByttKort4 = -1 Then
    Beep
Else ByttKort4 = -1
    Kort4.Cls
    KortFyra.Text = " "
    LäggUtKort Kort4,KortFyra,Färg4,Värde4
End If

End Sub

Sub Kort5_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)
If ByttKort5 = -1 Then
    Beep
Else ByttKort5 = -1
    Kort5.Cls
    KortFem.Text = " "
    LäggUtKort Kort5,KortFem,Färg5,Värde5
End If

End Sub

```

När användaren är klar med att byta kort klickar han/hon på kommandoknappen *Klar*. Handen värderas av proceduren *Jämför*. Variabeln "Spelare" håller reda på vilken av deltagarna som just har spelat. Poängtalet ska skrivas ut i rätt ruta, och summan av poängen räknas ihop med spelarens tidigare poäng.

```

Sub KlarKnp_Click()
    Dim Poäng%

```

## KAPITEL 16

```

jämför Färg1, Värde1, Färg2, Värde2, Färg3,
värde3, Färg4, Värde4, Färg5, Värde5, Hand$  

Select Case Hand  

Case "Par"  

    Poäng = 1  

Case "Två Par"  

    Poäng = 2  

Case "Triss"  

    Poäng = 3  

Case "Stege"  

    Poäng = 5  

Case "Färg"  

    Poäng = 7  

Case "Kåk"  

    Poäng = 10  

Case "Fyrtal"  

    Poäng = 15  

Case "Straight Flush"  

    Poäng = 20  

Case "Femtal"  

    Poäng = 25  

Case Else  

    Poäng = 0  

End Select  

Select Case Spelare  

Case "Spelare1"  

    Spelare1poäng = Spelare1poäng + Poäng  

    Spelare1.Text = Str$(Spelare1poäng)  

Case "Spelare2"  

    Spelare2poäng = Spelare2poäng + Poäng  

    Spelare2.Text = Str$(Spelare2poäng)  

Case "Spelare3"  

    Spelare3poäng = Spelare3poäng + Poäng  

    Spelare3.Text = Str$(Spelare3poäng)  

Case "Spelare4"  

    Spelare4poäng = Spelare4poäng + Poäng

```

```

    Spelare4.Text = Str$(Spelare4poäng)
End Select
Form3.ForeColor = RED
Form3.FontSize = 18
If Spelare1poäng >= Maxpoäng Then
    Form3.Show
    Form3.Print
    Form3.Print "Spelare 1 Vann!!"
ElseIf Spelare2poäng >= Maxpoäng Then
    Form3.Show
    Form3.Print "Spelare 2 Vann!!"
ElseIf Spelare3poäng >= Maxpoäng Then
    Form3.Show
    Form3.Print "Spelare 3 Vann!!"
ElseIf Spelare4poäng >= Maxpoäng Then
    Form3.Show
    Form3.Print "Spelare 4 Vann!!"
End If
End Sub

```

Proceduren Jämför kontrollerar de möjliga kombinationerna

```

Sub Jämför (F1$, V1%, F2$, V2%, F3$, V3%,
            F4$, V4%, F5$, V5%, Hand$)

```

## KAPITEL 16

```

If Femtal(V1, V2, V3, V4, V5) Then
    Hand = "Femtal"
ElseIf Flush(F1, F2, F3, F4, F5) And
Stege(V1, V2, V3, V4, V5) Then
    Hand = "Straight Flush"
ElseIf Fyrtal(V1, V2, V3, V4, V5) Then
    Hand = "Fyrtal"
ElseIf Kåk(V1, V2, V3, V4, V5) Then
    Hand = "Kåk"
ElseIf Flush(F1, F2, F3, F4, F5) Then
    Hand = "Färg"
ElseIf Stege(V1, V2, V3, V4, V5) Then
    Hand = "Stege"
ElseIf Triss(V1, V2, V3, V4, V5) Then
    Hand = "Triss"
ElseIf TvåPar(V1, V2, V3, V4, V5) Then
    Hand = "Två Par"
ElseIf Par(V1, V2, V3, V4, V5) Then
    Hand = "Par"
Else Hand = "Blank"
End If
End Sub

```

Här följer alla funktioner som testar respektive händer. Vi börjar med femtal och fyrtal för att se om vi har fem eller fyra av samma valör.

```

Function Femtal (V1 As Integer, V2 As
Integer, V3 As Integer, V4 As Integer, V5
As Integer) As Integer
If V1 = V2 AND V2 = V3 AND V3 = V4 AND V4 =
V5 Then
    Femtal = -1
    Else Femtal = 0
End If
End Function

Function Fyrtal(V1 As Integer, V2 As
Integer, V3 As Integer, V4 As Integer, V5
As Integer)
If V1 = V2 AND V2 = V3 AND V3 = V4 Then
    Fyrtal = -1
    ElseIf V2 = V3 AND V3 = V4 AND V4 = V5 Then
        Fyrtal = -1
    ElseIf V1 = V3 AND V3 = V4 AND V4 = V5 Then
        Fyrtal = -1
    ElseIf V1 = V2 AND V2 = V4 AND V4 = V5 Then
        Fyrtal = -1
    ElseIf V1 = V2 AND V2 = V3 AND V3 = V5 Then
        Fyrtal = -1
    Else Fyrtal = 0
End If
End Function

```

Funktionen Kåk undersöker om vi har en triss. Om vi har det kontrolleras om de återstående två är ett par.

```

Function Kåk (V1 As Integer, V2 As Integer,
V3 As Integer, V4 As Integer, V5 As
Integer) As Integer

```

## KAPITEL 16

```

If V1 = V2 And V2 = V3 And V4 = V5 Then
    Kåk = -1
ElseIf V1 = V2 And V2 = V4 And V4 = V5 Then
    Kåk = -1
ElseIf V1 = V2 And V2 = V5 And V3 = V4 Then
    Kåk = -1
ElseIf V1 = V3 And V3 = V4 And V2 = V5 Then
    Kåk = -1
ElseIf V1 = V3 And V3 = V5 And V2 = V4 Then
    Kåk = -1
ElseIf V1 = V4 And V4 = V5 And V2 = V3 Then
    Kåk = -1
ElseIf V2 = V3 And V3 = V4 And V1 = V5 Then
    Kåk = -1
ElseIf V2 = V3 And V3 = V5 And V1 = V4 Then
    Kåk = -1
ElseIf V2 = V4 And V4 = V5 And V1 = V3 Then
    Kåk = -1
ElseIf V3 = V4 And V4 = V5 And V1 = V2 Then
    Kåk = -1
Else Kåk = 0
End If

End Function

```

Funktionen *Flush* testar om alla fem färgerna är lika.

```

Function Flush (F1 As String, F2 As String,
F3 As String, F4 As String, F5 As String)
As Integer

If F1 = F2 AND F2 = F3 AND F3 = F4 AND F4 =
F5 Then
    Flush = -1
    Else Flush = 0
End If
End Function

```

Funktionen *Stege* kontrollerar om alla valörer är i en sekvens. Vi måste också tillåta sekvenser med esset i topp. Vi börjar med att sortera korten i storleksordning med proceduren *Sortera*. (*Sortera* finns definierad sist i det här kapitlet. Denna definition är definitivt inte rätt sätt att skriva den på, men eftersom vi inte gått igenom vektorer har vi varit tvungna att göra en mycket primitiv variant. För den som vill göra programmet själv kan det naturligtvis vara bra att ha tillgång till den, men som instruktivt exempel lämnar den mycket i övrigt att önska, och kan gärna ignoreras).

```

Function Stege (V1 As Integer, V2 As
Integer, V3 As Integer, V4 As Integer, V5
As Integer) As Integer

```

```

Dim S1%, S2%, S3%, S4%, S5%
Sortera V1, V2, V3, V4, V5, S1, S2, S3, S4,
S5
If S2 - 1 = S3 And S3 - 1 = S4 And S4 - 1 =
S5 Then
    If S1-1=S2 Or S2=13 And S1=1 Then
        Stege = -1
    Else Stege = 0
End If
Else Stege = 0
End If

End Function

```

Nästa funktion kontrollerar om vi har tre av samma valör

```

Function Triss (V1 As Integer, V2 As
Integer, V3 As Integer, V4 As Integer, V5
As Integer) As Integer

```

## KAPITEL 16

```
If V1 = V2 And V2 = V3 Then
    Triss = -1
ElseIf V1 = V2 And V2 = V4 Then
    Triss = -1
ElseIf V1 = V2 And V2 = V5 Then
    Triss = -1
ElseIf V1 = V3 And V3 = V4 Then
    Triss = -1
ElseIf V1 = V3 And V3 = V5 Then
    Triss = -1
ElseIf V1 = V4 And V4 = V5 Then
    Triss = -1
ElseIf V2 = V3 And V3 = V4 Then
    Triss = -1
ElseIf V2 = V3 And V3 = V5 Then
    Triss = -1
ElseIf V2 = V4 And V4 = V5 Then
    Triss = -1
ElseIf V3 = V4 And V4 = V5 Then
    Triss = -1
Else Triss = 0
End If
```

End Function

Funktionerna *TvåPar* och *Par* består även de av ett antal jämförelser. Kom ihåg att vi bara kommer att anropa *TvåPar* eller *Par* om alla tidigare tester misslyckats. Vi kan alltså räkna med att vi inte har något fyrtal eller någon triss.

```
Function TvåPar(V1 As Integer, V2 As
    Integer, V3 As Integer, V4 As Integer, V5
    As Integer) As Integer
```

```
If V1 = V2 Then
    If V3 = V4 OR V3 = V5 OR V4 = V5 Then
        TvåPar = -1
    ElseIf V1 = V3 Then
        If V2 = V4 OR V2 = V5 OR V4 = V5 Then
            TvåPar = -1
        ElseIf V1 = V4 Then
            If V2 = V3 OR V2 = V5 OR V3 = V5 Then
                TvåPar = -1
            ElseIf V1 = V5 Then
                If V2 = V3 OR V2 = V4 OR V3 = V4 Then
                    TvåPar = -1
                ElseIf V2 = V3 Then
                    If V4 = V5 Then TvåPar = -1
                ElseIf V2 = V4 Then
                    If V3 = V5 Then TvåPar = -1
                ElseIf V2 = V5 Then
                    If V3 = V4 Then TvåPar = -1
                Else TvåPar = 0
            End If
        End Function
Function Par(V1 As Integer, V2 As Integer,
    V3 As Integer, V4 As Integer, V5 As
    Integer) As Integer
    If V1 = V2 OR V1 = V3 OR V1 = V4 OR V1 = V5
    OR V2 = V3 OR V2 = V4 OR V2 = V5 OR V3 = V4
    OR V3 = V5 OR V4 = V5 Then
        Par = -1
    Else Par = 0
    End If
End Function
```

## KAPITEL 16

Sortera anropades i funktionen Stege för att sortera korten i nummerordning.

```
Sub Sortera (V1%, V2%, V3%, V4%, V5%, S1%,
S2%, S3%, S4%, S5%)
```

```
S1 = högst(V1, V2, V3, V4, V5)
S2 = nästhögst(V1, V2, V3, V4, V5, S1)
S5 = lägst(V1, V2, V3, V4, V5)
S4 = nästlägst(V1, V2, V3, V4, V5, S5)
S3 = trea(V1, V2, V3, V4, V5, S1, S2, S5,
S4)
```

```
End Sub
```

```
Function högst (T1%, T2%, T3%, T4%, T5%) As
Integer
```

```
If T1 >= T2 And T1 >= T3 And T1 >= T4 And
T1 >= T5 Then
    högst = T1
ElseIf T2 >= T1 And T2 >= T3 And T2 >= T4
And T2 >= T5 Then
    högst = T2
ElseIf T3 >= T1 And T3 >= T2 And T3 >= T4
And T3 >= T5 Then
    högst = T3
ElseIf T4 >= T1 And T4 >= T2 And T4 >= T3
And T4 >= T5 Then
    högst = T4
Else högst = T5
End If
```

```
End Function
```

```
Function lägst (T1%, T2%, T3%, T4%, T5%) As
Integer
```

```
If T1 <= T2 And T1 <= T3 And T1 <= T4 And
T1 <= T5 Then
    lägst = T1
```

```
ElseIf T2 <= T1 And T2 <= T3 And T2 <= T4
And T2 <= T5 Then
    lägst = T2
```

```
ElseIf T3 <= T1 And T3 <= T2 And T3 <= T4
And T3 <= T5 Then
    lägst = T3
```

```
ElseIf T4 <= T1 And T4 <= T2 And T4 <= T3
And T4 <= T5 Then
    lägst = T4
```

```
Else lägst = T5
```

```
End If
```

```
End Function
```

```
Function nästhögst (T1%, T2%, T3%, T4%,
T5%, hö%) As Integer
```

```
Select Case hö
```

```
Case T1
    nästhögst = nh(T2, T3, T4, T5)
```

```
Case T2
    nästhögst = nh(T1, T3, T4, T5)
```

```
Case T3
    nästhögst = nh(T1, T2, T4, T5)
```

```
Case T4
    nästhögst = nh(T1, T2, T3, T5)
```

```
Case T5
    nästhögst = nh(T1, T2, T3, T4)
```

```
End Select
```

```
End Function
```

## KAPITEL 16

```

Function nästlägst (T1%, T2%, T3%, T4%,
T5%, lä%) As Integer

Select Case lä
Case T1
    nästlägst = nl(T2, T3, T4, T5)
Case T2
    nästlägst = nl(T1, T3, T4, T5)
Case T3
    nästlägst = nl(T1, T2, T4, T5)
Case T4
    nästlägst = nl(T1, T2, T3, T5)
Case T5
    nästlägst = nl(T1, T2, T3, T4)
End Select

End Function

Function trea (T1%, T2%, T3%, T4%, T5%,
hö%, nhö%, lä%, nlä%) As Integer
If T1 <> hö And T1 <> nhö And T1 <> nlä And
T1 <> lä Then
    trea = T1
ElseIf T2 <> hö And T2 <> nhö And T2 <> nlä
And T2 <> lä Then
    trea = T2
ElseIf T3 <> hö And T3 <> nhö And T3 <> nlä
And T3 <> lä Then
    trea = T3
ElseIf T4 <> hö And T4 <> nhö And T4 <> nlä
And T4 <> lä Then
    trea = T4
Else trea = T5
End If

End Function

```

```

Function nh (T1%, T2%, T3%, T4%) As Integer
If T1 >= T2 And T1 >= T3 And T1 >= T4 Then
    nh = T1
ElseIf T2 >= T1 And T2 >= T3 And T2 >= T4
Then
    nh = T2
ElseIf T3 >= T1 And T3 >= T2 And T3 >= T4
Then
    nh = T3
Else nh = T4
End If

End Function

Function nl (T1%, T2%, T3%, T4%)
If T1 <= T2 And T1 <= T3 And T1 <= T4 Then
    nl = T1
ElseIf T2 <= T1 And T2 <= T3 And T2 <= T4
Then
    nl = T2
ElseIf T3 <= T1 And T3 <= T2 And T3 <= T4
Then
    nl = T3
Else nl = T4
End If

End Function

Slutligen kommer vi till när användaren tröttnat och trycker på
Slut knappen.

Sub Slut_Click
    End
End Sub

```

KUNGL  
BIBLIOTEKET  
STOCKHOLM

CHRISTER STURMARK

Lär dig använda

# VISUAL BASIC

ALMQVIST & WIKSELL / DATAFÖRLAGEN

000766961

01

häck 12834



Christer [S] NY LÄNTAGARE!

2025-11-07

Lär dig använda **Visual Basic** är en bok för dig som vill lära dig att skriva program för Windows med programmeringsverktyget Microsoft Visual Basic.

Microsoft Visual Basic är lätt att lära sig. Även de som har mycket liten programmeringserfarenhet kan snabbt lära sig att skriva avancerade Windows-program med fönster, menyer, knappar etc. Boken visar vilka stora möjligheter som finns i en grafiskt objektorienterad miljö jämfört med den traditionella och visar med exempel och övningar hur du snabbt bygger snygga och avancerade applikationer.

Författaren **Christer Sturmark** är "Windowswatcher" och grundare av DataMedia, ett utbildnings- och informationsföretag som uteslutande arbetar med Windows och Windowsprogram. Christer är en erfaren utbildare och har skrivit ett flertal böcker om Windows och Windowsprogram.

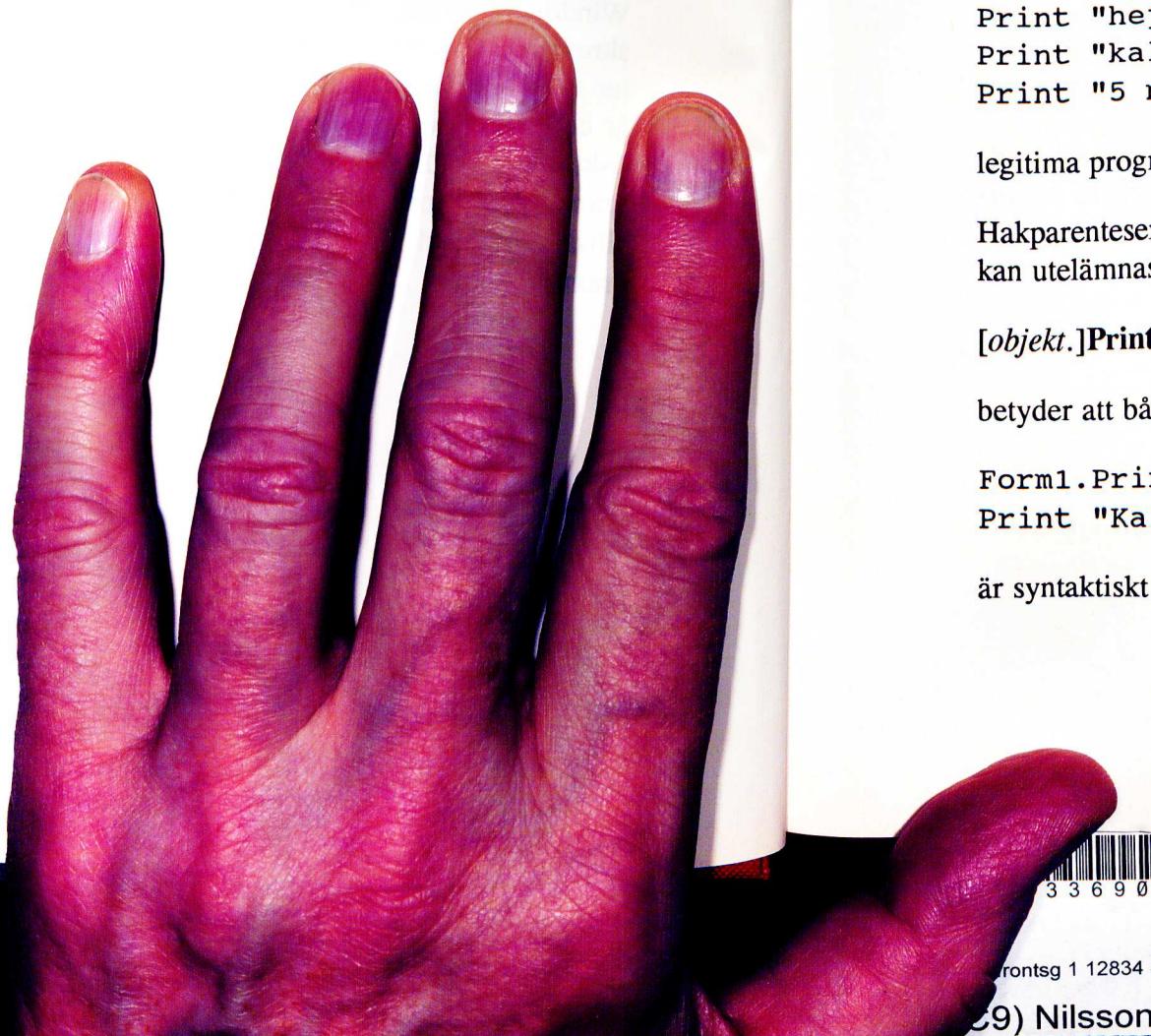
Best nr 21-12782-4  
Art nr 21-12782-4

**Almqvist & Wiksell**  
DATAFÖRLAGEN

Almqvist & Wiksell Förlag AB, Tfn 0470-166 20  
Almqvist & Wiksell Förlag AB, Tfn 08-734 30 00

Jag vill passa på att tacka min mycket kompetenta medförfattare Amelie Banks för hennes insats. Det har varit roligt att arbeta med detta projekt tillsammans med henne. Hon var en gång i tiden min lärare vid Uppsala Universitet, då jag läste Artificiell Intelligens där. Jag vill också passa på att tacka det företag som DataMedia varit "sambo" med ett par år, nämligen Culmen Gruppen, med Mats Israelsson och Kristina Lund-Zenzén i spetsen. Det har varit en mycket trevlig tid och vi är mycket tacksamma för all hjälp vi har fått.

Stockholm den 20 Jan 1992  
Christer Sturmark, VD DataMedia AB



# Inledning

Innan vi börjar med själva programmet ska vi förklara en del skrivsätt som kommer att användas genomgående i boken.

Ibland ger vi ett syntaxmönster för ett kommando. Den text som står i **fetstil** ska upprepas bokstavligen. Den text som står i *kursiverad* stil visar vilken typ av objekt som ska in på den platsen.

Exempel: Syntaxen för kommandot *Print* är

**Print sträng**

Vilket betyder att man ska skriva ordet "Print" följt av en sträng. En sträng är detsamma som en teckenföljd. Exempelvis är

```
Print "hej"  
Print "kalle är dum"  
Print "5 myror är fler än 6738%&&//"
```

legitima programsatser.

Hakparenteser betyder att det som står innanför parenteserna kan utelämnas.

[*objekt.*]Print sträng

betyder att båda dessa alternativ:

```
Form1.Print "Kalle"  
Print "Kalle"
```

är syntaktiskt riktiga.