

Introduksjon til Visual Basic 5

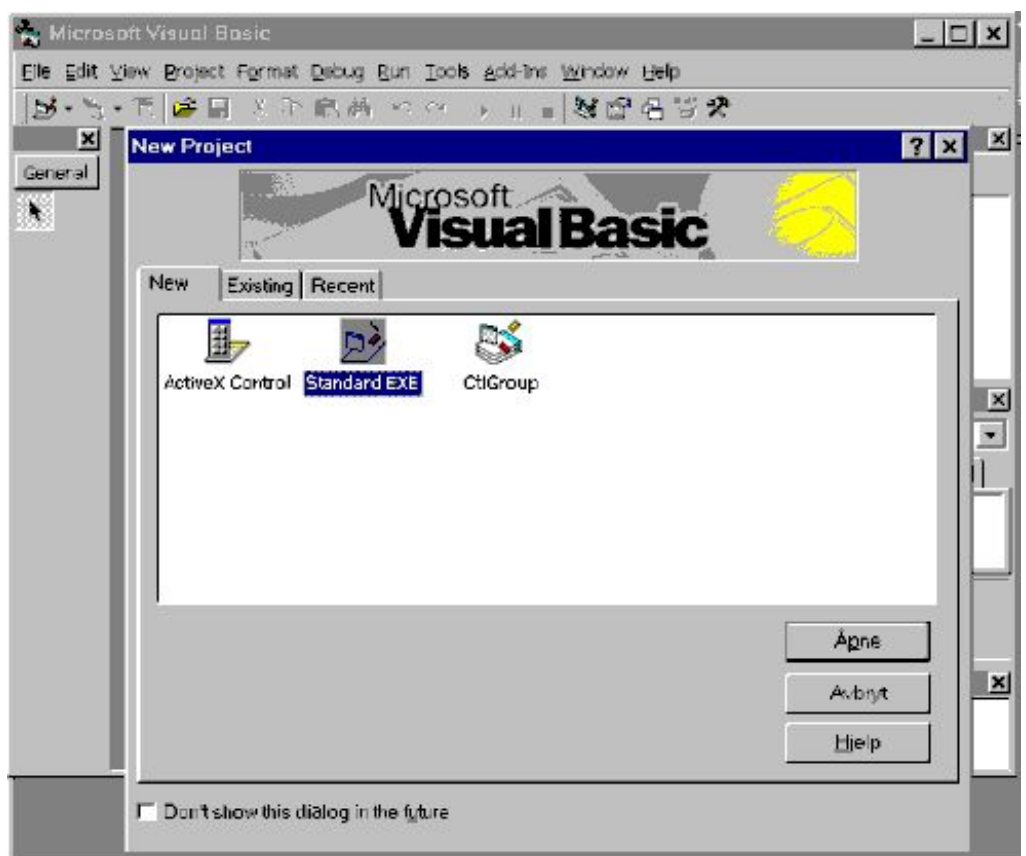
Denne håndboken gir en kort og enkel innføring i programmering i Visual Basic ver. 5. Her er CCE versjonen brukt, Microsoft la den ut for nedlasting på sin webserver. Fila VB5CCEIN.EXE installerer VB 5 CCE på datamaskinen. Fila VB5CCEHL.EXE installerer hjelpesystemet (kjør denne etter installasjonen).

Hjelpesystemet er bra, og det henvises til det for hjelp med mer spesielle rutiner. Her gis det bare en enkel innføring i sekvenser, valg, løkker og filbehandling.

Visual Basic er objektbasert, dvs. at de elementene (objektene) som bygger opp programmene dine inneholder både informasjon (data) og rutiner som bearbejder informasjonen. Hvert objekt har en rekke egenskaper ("Properties") som du kan endre, i tillegg kan du skrive inn programrutiner som endrer disse egenskapene.

Oppstart

Normalt vil installasjonsfila legge inn en snarvei, slik at du kan starte VB via **Start** og **Programmer**. Du kan også bruke Windows Utforsker og dobbeltklikke på fila *VB5CCE.EXE* i katalogen der programmet er installert (som oftest Programfiler\Vb5cce). Du får da fram et skjerm- bilde omtrent som det vist til høyre. For å starte et nytt programmeringsprosjekt velger du arkfanen **New** og **Standard EXE**. Klikk så på **Åpne** knappen for å opprette et nytt Visual Basic prosjekt.

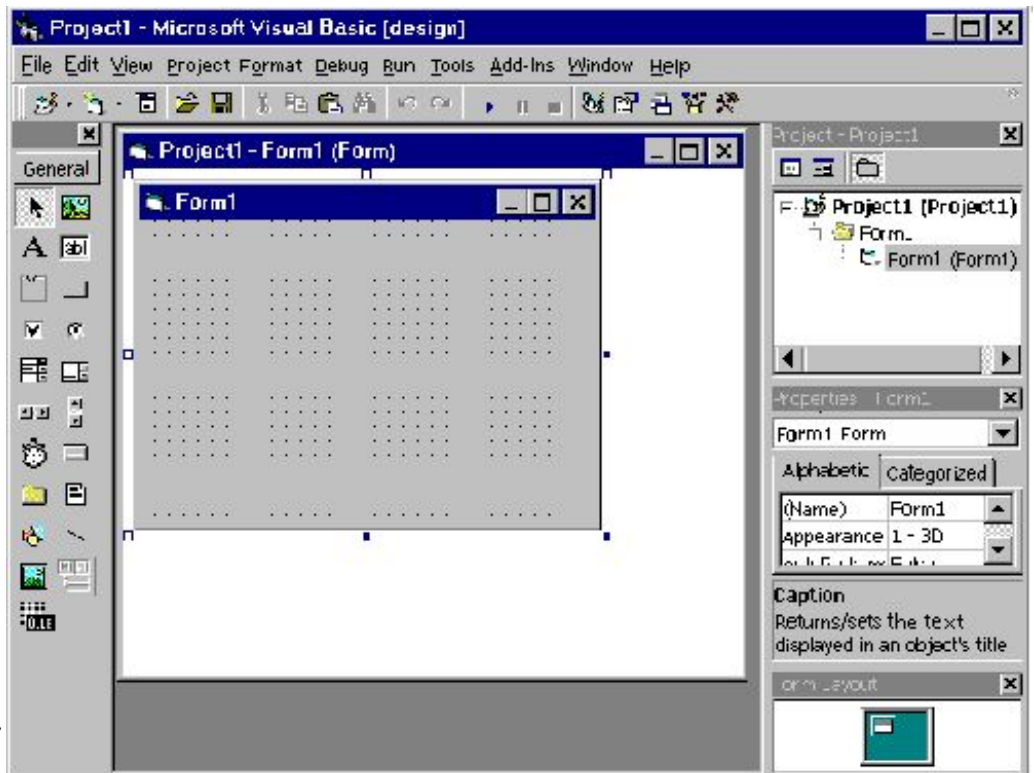


Du får da fram et skjermbilde omtrent som vist til høyre.

Til venstre (under *General*) finner du verktøyboksen. Den inneholder objekter (elementer) du kan plassere i formvinduet (*Form1*). Disse objektene bruker du til å lage det visuelle bruker-grensesnittet. Selve formvinduet blir hovedvinduet til det ferdige programmet ditt.

Til høyre finner du egenskaps-editor ("Properties"), her endrer du egenskapene til objektet (f.eks. navn, tittel ("Caption")) osv. Dersom vinduet ikke finnes, trykker du på funksjonstast **F4**.

Over egenskaps editor finnes prosjektvinduet, her ser du alle moduler som prosjektet inneholder. Dobbeltklikk på den modul du ønsker å arbeide med.



Vårt første program

Start VB og start et nytt prosjekt (**File, New Project**). De fleste innledende programmer består i å skrive ut en beskjed på skjermen, f.eks. Hei. Alle VB programmer starter med et vindu ("Form") som automatisk blir gitt navnet *Form1*, mens prosjektet får navnet *Project1*. Du kan endre tittelen til vinduet fra egenskapsvinduet ("Properties"). Finn egenskapen "Caption" til Form1, klikk til høyre for den og skriv inn f.eks. "Test1". Den enkleste måten å få skrevet ut tekst er å bruke en etikett ("Label"). Klikk på **A** i verktøykassen ("Toolbox") og dra ut et passende rektangel øverst i vinduet (Form1). I

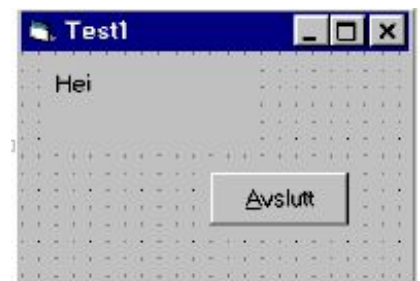
egenskap vinduet finner du egenskapen "Caption", klikk til høyre for den og skriv inn "Hei" isteden for "Label1" som VB automatisk legger inn. Du kan nå prøve å kjøre programmet. Velg **Run, Start** (el. trykk **F5**).

Vi ønsker også en kommandoknapp som avslutter programmet når vi klikker på den. Velg "CommandButton" i verktøyboksen og dra ut en passende størrelse nede til høyre i vinduet, se figuren. Sett egenskapen "Name" til *CmdAvslutt*, og "Caption" til *&Avslutt*. "&" tegnet fører til at A blir understreket, det betyr at du kan trykke ALT A tastekombinasjonen for å avslutte programmet.

Det eneste som nå mangler er den programkoden som skal utføres når kommandoknappen trykkes. For å lage den kan du dobbeltklikke på kommandoknappen, da legger VB inn en innledningslinje og en avslutningslinje med en tom linje mellom. I den tomme linjen skriver du inn *End* som angir at programmet skal avsluttes, slik:

```
Private Sub ExitBtn_Click()
    End ' Avslutt programmet.
End Sub
```

' *Avslutt programmet* er en kommentar som ignoreres av VB. Den enkle apostrofen innleder en kommentar (som varer til slutten av linja). Det er lurt å legge inn kommentarer som forteller hva de ulike



instruksjonene/rutinene gjør. Da blir det mye lettere å forstå programmet når du kikker på det ved en senere anledning.

Lagre

Dersom du ønsker å lagre testprosjektet kan du velge **File, Save Project As**. Det er alltid lurt å lagre arbeidet før du kjører programmet.

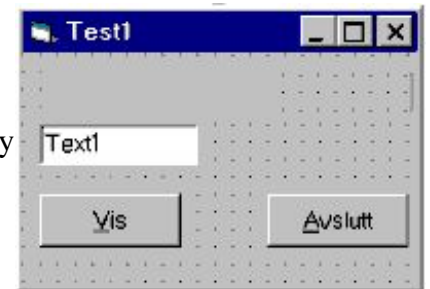
Deretter velger du **Run, Start** for å kjøre programmet. Vinduet vårt kommer opp på skjermen, og teksten vises. Klikker du på "Avslutt" knappen eller ALT A på tastaturet så avsluttes programmet vårt.

Inndata

La oss lage et program der bruker kan taste inn en melding, trykke på en knapp og vise meldingen. Gjentas dette vil en teller vises. Du kan bruke vårt "Test1" prosjekt. Slett egenskapen "Caption" til etiketten (slett Hei).

Legg til en tekstboks ("TextBox" fra verktøyboksen) under etiketten, og en ny trykkknapp nede til venstre. Sett egenskapen "Caption" til *Vis*, og egenskapen "Name" til *CmdVis*. Se figuren til høyre.

For å legge inn programkoden til "Vis" knappen dobbeltklikker du på den og legger inn koden som vist under.



I tillegg trenger vi et par variabler, de deklarerer med en *Dim* setning.

```
' Deklarer globale variabler:
Dim Antall As Integer          ' Deklarer Tallvariabel for hele formvinduet

Private Sub CmdVis_Click()
    Dim InnText As String      ' Deklarer lokal Strengvariabel
    ' Kopier Text1 til InnText når Knapp trykkes.
    InnText = Text1.Text
    Antall = Antall + 1        ' Hvor mange trykk?
    Label1.Caption = InnText + Str$(Antall) ' Vis InnText
End Sub

Private Sub CmdAvslutt_Click()
    End ' Avslutt programmet.
End Sub
```

Her deklarerer variablene "Antall" utenfor alle prosedyrer i programmet. Det betyr at variabelen er synlig for alle prosedyrer i dette formvinduet (men ikke i andre). Variablen "InnText" er en lokal variabel siden den er deklarerert inni en prosedyre, og den er kun synlig inni denne prosedyren.

Variablen InnText tilordnes verdien til tekstboksen "Text1" sin "Text" egenskap (*Text1.Text*). Deretter økes telleren "Antall" med en. Til slutt får "Label1.Caption" verdien til variabelen "InnText" + antall ganger "Vis" knappen er blitt trykket. For å omforme tallvariablen "Antall" til en tekststreng bruker vi funksjonen *Str\$*.

Variabler og datatyper

Enkelt kan vi si at en variabel er et område i datamaskinens hukommelse som vi får adgang til via et navn (navnet utgjør adressen). Vi kan endre variabelens innhold, og vi kan lese innholdet.

En variabel deklarerer (opprettet) med det reserverte ordet *Dim*, så følger navnet på variabelen, det reserverte ordet *As* og variabelens datatype (se nedenfor):

```
Dim x As Integer
```

Navnet må begynne med en bokstav, så bokstaver, tall eller understreksymbolet (`_`).

Tilordningssymbolet "=" brukes for å gi variabelen en verdi. En variabel har også en datatype som forteller hvilke verdier som kan lagres (tilordnes) i variabelen, og hvilke operasjoner som den kan delta i. Her kommer noen av datatypene som Visual Basic bruker:

Datatype	Verdiområde	Antall byte
Byte	0 – 255	1
Integer	Heltall +- 32767	2
Long	Heltall +- 2mrd.	4
Single	Desimaltall	4
Double	Store Desimaltall	8
String	Tekststreng	Varies

Valg

I de små eksempler vi har sett på til nå har programmet først utført linje en, så linje to osv. Vi kaller en slik programflyt for en *sekvens*.

Imidlertid har vi ofte behov for at programflyten tar en av to mulige retninger – *valg*. La oss se på et lite eksempel. Start et nytt prosjekt (**File, New Project**), sett formvinduets "Caption" til *Valg*, og "AutoRedraw" til *True*. Nederst til høyre plasseres en trykknapp ("Name" settes til *CmdBeregn*, "Caption" til *&Beregn*). Over plasseres to tekstbokser med passende ledetekst, se figuren til høyre.



Dobbelklikk på "Beregn" knappen for å få fram kodevinduet, skriv inn så koden blir slik:

```
Private Sub CmdBeregn_Click()
    Dim x As Integer, y As Integer
    Dim Produkt As Integer, Kvotient As Single
    x = Val(Text1.Text)           ' Inndata
    y = Val(Text2.Text)

    Produkt = x * y               ' Beregninger
    Kvotient = x / y
    Cls
    Print x; " * "; y; " = "; Produkt ' Utdata
    Print x; " / "; y; " = "; Kvotient
End Sub
```

Lagre programmet (med f.eks. filnavnet Kalkulator), og kjør programmet og kontroller at beregningene er korrekte.

Programmet inneholder tre hoved deler; en *Inndatadel*, en *Bergningsdel* og en *Utdatadel*. De fleste programmer vi skal lage vil være bygget opp på denne måten.

Linjen

```
x = Val(Text1.Text)
```

omformer det som skrives i tekstboksen til en tallverdi (dersom det er mulig) som plasseres i (dvs. tilordnes) variabelen "x". En tekstboks returnerer alltid en strengverdi, så vi bør alltid omforme tekststrengen til et tall med *Val* funksjonen. Så beregnes produkt og kvotient og resultatene vises på skjermen med en "Print" setning. Vi skiller de ulike variablene som skal skrives ut med semikolon. Semikolon fører til at variablene skrives ut uten mellomrom mellom dem.

Hva skjer dersom du taster inn verdien null for Tall 2? Programmet genererer en kjøretidsfeil fordi divisjon med null ikke er definert. Vi må kontrollere om bruker gir Tall 2 verdien null, dersom det er tilfelle må vi vise en feilmelding på skjermen. Ellers utfører vi beregningen og viser resultatene på skjermen.

For å gjøre et slikt valg bruker Visual Basic *If* setningen. Gå til kodevinduet og endre koden slik:

```
Private Sub CmdBeregn_Click()  
    Dim x As Integer, y As Integer  
    Dim Produkt As Integer, Kvotient As Single  
    x = Val(Text1.Text)           ' Inndata  
    y = Val(Text2.Text)  
  
    Produkt = x * y                ' Beregninger  
    If y <> 0 Then  
        Kvotient = x / y  
    Else  
        MsgBox "NB. Divisjon med null er ikke definert!"  
        Kvotient = -1              ' Signaliser feil  
    End If  
    Cls  
    Print x; " * "; y; " = "; Produkt ' Utdata  
    Print x; " / "; y; " = "; Kvotient  
End Sub
```

If setningen har et logisk uttrykk: *If y <> 0*. Dersom dette uttrykket er sant utføres beregningen, ellers vises setningene etter *Else*, dvs. en meldingsboks på skjermen, og Kvotient får verdien -1 for å vise at dette er en feil. *If* setningen avsluttes med *End If*.

Oppg.

Utvid eksemplet ovenfor slik at programmet også beregner summen av tallene og differansen.

Mer om meldingsboksen

Meldingsboksen som ble brukt i Valg programmet var meget enkel. Meldingsboksen er en del av Windows operativsystemet, og kan brukes både som en prosedyre (som ovenfor), og som en funksjon. En funksjon vil alltid returnere en verdi (og plasseres derfor til høyre for tilordningssymbolet =), mens en prosedyre ikke gjør det. Den generelle syntaksen for en meldingsboks er slik:

```
Msgbox (Meldingstekst, meldingstype, overskrift)
```

Introduksjon til Visual Basic 5 (TH)

Vi kunne ha skrevet meldingen slik:

```
MsgBox "NB. Divisjon med null er ikke definert!", vbOKOnly, "Feil"
```

Men meldingsboksen har flere muligheter, når en feil oppstår vil vi gjerne gi brukeren en mulighet for å rette opp feilen. Vi bruker da meldingsboksen som en funksjon (parametrene plasseres inni en parentes), og lar en variabel ta imot det resultat som meldingsboksen returnerer.

Dersom du ønsker flere opplysninger om meldingsboksen, så skriver du ordet *Msgbox* i kode editoren, plasseres markøren i ordet og trykker på funksjonstast **F1**.

Vi kan velge mellom flere meldingstyper:

Meldingstype	Verdi
Vis bare OK knappen	vbOkOnly
Vis OK og Avbryt	vbOkCancel
Vis Avbryt, Prøv igjen, Ignorer	vb AbortRetryIgnore
Vis Ja og Nei	vbYesNo
Vis Ja, Nei, Avbryt	vbYesNoCancel

Den verdien som meldingsboksen returnerer avhenger av hvilken knapp bruker klikket på:

Returverdi	Valgt knapp
vbOk	Ok
vbCancel	Avbryt
vbAbort	Avbryt
vbRetry	Prøv igjen
vbYes	Ja
vbNo	Nei

Her er et eksempel:

```
Dim Ok As Integer
Ok = MsgBox("Feil inndataverdi. Prøve igjen?", vbYesNo, "Feil")
If Ok = vbYes Then
    ' Bruker valgte Ja
Else
    ' Bruker valgte Nei
End If
```

I tillegg kan vi utruste meldingsboksen med et ikon. Det vi gjør da er å legge til en tallkonstant til meldingstypen. Du kan bruke disse tallkonstantene (se hjelp for *Msgbox*):

Tallkonstant	Ikon
vbCritical	Kritisk melding
vbQuestion	Advarselspørring
vbExclamation	Utropstegn

vbInformation	Informasjon
---------------	-------------

Her er eksemplet ovenfor utvidet slik at meldingsboksen viser ikonet advarselspørring:

```
Ok = MsgBox("Feil inndataverdi. Prøve igjen?", vbYesNo + vbQuestion, "Feil")
```

Oppg.

Kalkulatorprogrammet mangler en "Avslutt" knapp. Legg til en slik knapp på formvinduet. La koden som utføres når knappen klikkes spørre brukeren om det virkelig er meningen å avslutte programmet (bruk en meldingsboks med Ja og Nei knapp). Bare dersom brukeren svarer Ja avsluttes programmet.

Formattering av tall

I eksempelet ovenfor beregnet vi kvotienten av det ene tallet dividert med det andre. Men vi kunne ikke angi hvor mange desimaler svaret skulle vises med. Også dersom flere tall skal vises underhverandre er det vanlig at enere, tiere osv. plasseres under hverandre. Det kan vi få til ved å bruke *Format\$* funksjonen. La oss si at variabelen x inneholder resultatet av divisjonen $1/3$, og at vi ønsker resultatet vist med en desimal. Det kan vi gjøre slik:

```
Print Format$(x, "0.0")
```

Den andre parameteren til funksjonen ("0.0" – formatstrengen) angir at verdien til variabelen skal skrives med ett desimalsiffer. Dersom resultatet blir mindre enn null, skal en innledende 0 skrives foran desimaltegnet. Legg merke til at *Format\$* funksjonen bruker punktum som desimalskilletegn (VB er et amerikansk program).

Dersom x inneholder en større verdi (som 12643.3333) og vi ønsker x utskrevet med to desimaler, og tusenskilletegn, blir formatstrengen slik:

```
Print Format$(x, "#,##0.00")
```

Her vil # tegnet i formatstrengen angi at dersom det er siffer i denne posisjonen, så skrives sifferet, ellers skrives et blankt tegn.

I formatstrengen har disse tegn betydning:

angir at eventuelt siffer skal skrives, ellers skrives blankt

0 angir at eventuelt siffer skrives, ellers skrives 0

. angir hvor vi ønsker desimalskilletegnet

, angir at vi ønsker store tall skrevet med tusenskilletegn

Løkker (gjentakelse)

I mange sammenhenger har vi behov for å gjøre den samme operasjonen flere ganger. La oss si at vi ønsker å lage et program som beregner gjennomsnittshøyden til alle elevene i klassen. Vi kunne deklarere 28 variabler, og lage et formvindu med 28 tekstbokser der elevenes høyde kunne registreres. Dette blir svært tungvint. Isteden bruker vi en *For* løkke og en inndataboks ("InputBox").

Start et nytt prosjekt, sett egenskapen "AutoRedraw" til *True*. Dobbelklikk i formvinduet slik at VB legger opp *Form_Load* prosedyren, og skriv inn denne koden:

```
Private Sub Form_Load()  
    Dim x As Integer, i As Integer, Sum As Long  
    Dim Snitt As Single
```

Introduksjon til Visual Basic 5 (TH)

```
For i = 1 To 5                                ' Inndata
  x = Val(TextBox("Oppgi høyde", "Høyde?"))
  Sum = Sum + x
Next i
Snitt = Sum / 5                                ' Beregn
Print "Gjennomsnittshøyden: "; Str$(Snitt) ' Utdata
End Sub
```

Inndataboksen (*TextBox*) er en funksjon som viser en dialogboks på skjermen. I denne boksen kan brukeren skrive inn en verdi, og så klikke på "OK" knappen, da lukkes boksen og den inntastede verdien tilordnes variabelen på venstre side av tilordningssymbolet (=). Siden inndataboksen returnerer en tekststreng, bruker vi *Val* funksjonen for å omforme verdien til et tall som tilordnes heltallsvariabelen x. Til slutt beregnes gjennomsnittshøyden og resultatet vises på skjermen.

Foruten *For* løkken har vi *Do .. Loop Until* løkken og *While ... Wend* løkken. Her kommer et par eksempler:

```
Dim Tall As Integer
Do
  Tall = Val(TextBox("Oppgi tall", "Tall (neg. avslutter)?"))
Until Tall < 0
. . .
Tall = 0
While Tall >= 0
  Tall = Val(TextBox("Oppgi tall", "Tall (neg. avslutter)?"))
Wend
. . .
```

I det første eksemplet testes verdien i slutten av løkka, i det andre eksemplet testes verdien i begynnelsen. Det betyr at i *While* løkka må vi passe på at Tall ikke har en negativ verdi, ellers utføres ikke løkka i det hel tatt!

Opgg.

Lag et enkelt program som summerer en rekke tall. Vi vet ikke på forhånd hvor mange tall som skal summeres. Når bruker taster inn tallet null avsluttes summeringen og summen vises på skjermen.

Bruk av flere vinduer

I litt større programmer bruker vi ofte et vindu som hovedmeny. Dette vinduet inneholder en rekke knapper. Når bruker trykker på en av disse, skjules hovedvinduet, og et nytt vindu (ny "Form") vises. Et vindu kan brukes til inntasting av data, et annet til å vise resultatet osv.

Start et nytt prosjekt (**File, New Project**), bruk egenskapsvinduet og sett egenskapen "Name" til *FrmMain* (hovedvinduet vårt) og egenskapen "Caption" til *Flere vinduer*. I hovedvinduet legger du inn to trykknapper. Sett egenskapen Caption til den øvre knappen til *&Vindu 2*, og "Name" til *CmdVindu2*. Den nedre knappen: *&Avslutt* og *cmdAvslutt*.

Når brukeren trykker på "Vindu 2", skal det dukke opp et nytt vindu ("Form") på skjermen. Velg **Project, Add Form** og legg til en ny "Form". Endre egenskapen "Name" til *FrmVindu2* og sett "Caption" til *Vindu 2*.



Plasser en trykknapp nederst, sett knappens "Caption" til *Lukk*, og "Name" til *CmdLukk*. Dobbelklikk på *Lukk* knappen og skriv inn:

```
FrmVindu2.Hide
```

slik at prosedyren blir slik:

```
Private Sub Command1_Click()  
    FrmVindu2.Hide    ' Skjul dette vindu  
    Form1.Show        ' Vis hovedvinduet  
End Sub
```

Legg merke til at vi bruker navnet på vinduet ("FrmVindu2") dvs. objektnavnet før navnet på operasjonen. Vi skiller de to med punktum.

Lagre med å velge File, Save FrmVindu2 As og gi den f.eks. navnet Test3F2.

Bruk Prosjekt vinduet (hvis det ikke vises velg View, Project Explorer) og velg hovedvinduet vårt – FrmMain – ved å dobbeltklikke på det. For å legge inn koden som skal utføres når trykknappen "Vindu 2" velges, dobbeltklikk på den. Visual Basic viser kodevinduet, skriv inn slik at prosedyren blir slik:

```
Private Sub CmdVindu2_Click()  
    Form1.Hide        ' Skjul dette vinduet  
    FrmVindu2.Show    ' Vis vindu 2  
End Sub
```

"Avslutt" knappen sin kode blir som tidligere:

```
Private Sub CmdAvslutt_Click()  
    End  
End Sub
```

Kjør programmet for å teste det.

Overføre informasjon mellom ulike vinduer

Dersom det ene vinduet brukes til innlesing av informasjon fra brukeren, mens et annet vindu beregner og viser resultatet, må denne informasjon kopieres til en global variabel i en modul (.BAS fil) (velg **Project, Add Module**). Globale variabler deklarerer ikke med Dim, men med det reserverte ordet **Global** el. **Public**. Global brukes oftest når en konstant defineres:

```
Global Const Pi = 3.14
```

Public brukes når variabler deklarerer (istedet for Dim), og for prosedyrer og funksjoner.

Oppg.

Vi introduserte løkker med et eksempel som beregnet gjennomsnittshøyden på elevene. Endre dette eksempelet slik at hovedvinduet inneholder tre knapper, den første starter en løkke som leser inn høyden til elevene, den andre viser et nytt formvindu der følgende informasjon vises:

Antall elever

Elevenes gjennomsnittshøyde

Dette formvinduet må ha en knapp som lukker vinduet og viser hovedvinduet.

Den tredje knappen i hovedvinduet brukes til å avslutte programmet.

Egendefinerte prosedyrer og funksjoner

Det er enklere å løse flere små problemer enn et stort. Derfor deler vi opp et prosjekt i flere deler – moduler. I Visual Basic er disse modulene prosedyrer ("Sub") og funksjoner ("Function"). Prosedyrer og funksjoner kalles ved at vi skriver navnet på dem i en Visual Basic setning. Vi har allerede brukt noen av Visual Basic sine innebygde funksjoner – *Val()*, *Str\$()* og *Format\$()*. Inni parenteser plasserer vi parametrene til funksjonen, f.eks.: *Val(Tekst)*. Her er variabelen *Tekst* en innparameter til funksjonen. Noen funksjoner kan ta flere parametre, da skilles parameternavnene med komma.

Forskjellen mellom prosedyre og funksjon er:

- Prosedyrer returnerer ikke en verdi, mens funksjoner alltid returnerer en verdi. Derfor plasseres funksjonskallet til høyre for tilordningssymbolet.
- I Visual Basic plasseres parametrene til et funksjonskall inni parentes, mens i prosedyrekall brukes ikke parentes.

Syntaksen for å deklarere en egendefinert prosedyre er slik (prosedyren defineres i en "Form

```
Private Sub MinProsedyre()  
    ' prosedyrens setninger  
End Sub
```

Vi kaller prosedyren slik (prosedyrekall):

```
MinProsedyre
```

Dersom den egendefinerte prosedyren med innparameter deklarerer slik:

```
Private Sub MinProsedyre(ByVal Antall As Integer)  
    ' prosedyrens setninger  
End Sub
```

Vi kaller en prosedyre med parametre slik:

```
MinProsedyre 4
```

Funksjoner deklarerer på omtrent samme måte som prosedyrer:

```
Private Function MinFunksjon(ByVal x As Integer) As Integer  
    ' funksjonens setninger, f.eks:  
    MinFunksjon = x * x    ' Angi funksjonens returverdi  
End Function
```

Siden en funksjon alltid returnerer en verdi, plasseres funksjonskallet til høyre for tilordningssymbolet (=), slik:

```
Kvadratet = MinFunksjon(2)
```

Dersom du har behov for flere parametre, skilles de med komma, f.eks. slik:

```
Private Function MinFunksjon(ByVal x As Integer,  
                             ByVal y As Integer) As Integer
```

"Public" prosedyrer/funksjoner

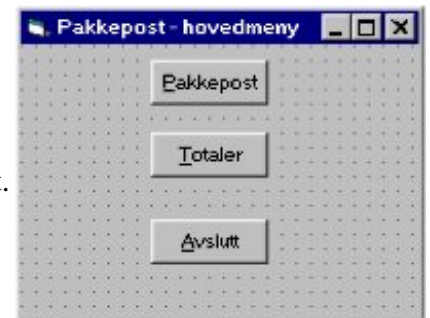
Dersom du deklarerer prosedyrer el. funksjoner i en modulfil (filtype .BAS) så bytter du ut ordet *Private* med *Public*.

Fordelen med å bruke en modulfil er at de variabler, funksjoner og prosedyrer som er deklarerert som *Public* i en modul vil være tilgjengelige for rutinene i alle formvinduerne i prosjektet vårt. I større programmeringsprosjekter brukes flere formvinduer, f.eks.:

- Et hovedvindu som utgjør menyen
- Ett vindu for inntasting av informasjon
- Ett vindu som viser resultatene

Når informasjonen som er hentet inn i Inndatavinduet lagres i variabler som er deklarerert som *Public* i prosjektets modulfil, vil denne informasjonen uten videre kunne brukes av Utdatavinduet.

La oss se på et lite eksempel. Vi skal lage et lite program som postkontoret kan bruke til å få oversikt over portoene som kundene skal betale for pakkepost. La oss for enkelhets skyld si at vi kun har to vektgrupper: under 1 kg (porto 15.-), og pakker på 1 kg el. mer (med en porto på 45.-). En kunde må kunne levere inn mer enn en pakke, og selvsagt av ulik vekt. Til slutt vises den totale pris kunden skal betale.



Posten ønsker også oversikt over hvor mange pakker av hvert slag som leveres inn hver dag, og hvor mye porto som er innbetalt for pakkepost i løpet av en dag.

Programmet skal ha et hovedvindu med menyvalgene:

Pakkepost, Totaler, Avslutt.

Se figuren til høyre. Når du lagrer prosjektet kan du lagre hovedvinduet som *PakkePost1.frm*.

Dobbelklikk på hver av kommandoknappene og legg inn denne programkoden:

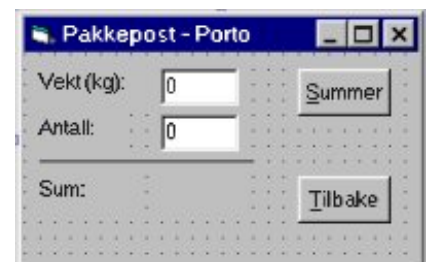
```
Private Sub CmdPakkePost_Click()
    Form1.Hide
    Form2.Show
End Sub

Private Sub CmdTotaler_Click()
    Form1.Hide
    Form3.Show
End Sub

Private Sub CmdAvslutt_Click()
    End
End Sub
```

Velg så **Project, Add form (New)** for å legge til ett nytt formvindu til prosjektet vårt. Legg til etiketter, inndata felt og trykknapper som figuren til høyre viser.

Legg inn denne koden for de to kommandoknappene:



```
Dim Sum As Single
```

```
Private Sub Form_Activate()
```

Introduksjon til Visual Basic 5 (TH)

```
Sum = 0
End Sub

Private Sub CmdSummer_Click()
    Dim Porto As Single, Vekt As Single, Ant As Integer
    Vekt = Val(Text1.Text)
    Ant = Val(Text2.Text)
    If Vekt < 1 Then
        Porto = 15 * Ant
        AntU1kg = AntU1kg + 1
    Else
        Porto = 45 * Ant
        AntO1kg = AntO1kg + 1
    End If
    Sum = Sum + Porto
    LblSum.Caption = Format$(Sum, "0.00")
    SluttSum = SluttSum + Porto
End Sub

Private Sub CmdTilbake_Click()
    Form2.Hide
    Form1.Show
End Sub
```

Siden posten ønsker en oversikt over det totale antall pakker av hvert slag, samt sluttsum for alle pakker har vi behov for tre globale variabler (AntU1kg, AntO1kg, Sluttsum). Disse må vi deklare i en egen modul. Velg **Project, Add Module (New)** og skriv inn denne koden:

```
' ----- Globale variabledeklarasjoner -----
Public AntU1kg As Long, AntO1kg As Long
Public SluttSum As Single
```

Siden vi ønsker at alle tall skal skrives ut under hverandre trenger vi en funksjon som returnerer tall formattert med to desimaler, og innledende nuller (f.eks. slik: 00.150,50). Denne funksjonen (*Formatter*) skriver du inn under de globale variabeldeklarasjonene i modulen som da blir slik:

```
' ----- Globale variabledeklarasjoner -----
Public AntU1kg As Long, AntO1kg As Long
Public SluttSum As Single

Public Function Formatter(ByVal Verdi As Single) As String
    Dim s As String
    s = Format$(Verdi, "00,000.00")
    Formatter = s
End Function
```

Til slutt legger du til et tredje formvindu, setter egenskapen "AutoRedraw" til *True* og legger til en "Tilbake" knapp. Dobbelklikk i formvinduet og legg inn kode for hendelsesprosedyren *FormActivate*:

```
Private Sub Form_Activate()
    Print "Antall pakker /u 1kg: "; Str$(AntU1kg)
    Print "Antall pakker /o 1kg: "; Str$(AntO1kg)
    Print "Total sum pakkeporto: "; Formatter(SluttSum)
End Sub

Private Sub CmdTilbake_Click()
    Form3.Hide
```

```
Form1.Show
End Sub
```

Prosedyre/Funksjons parametere

En parameter brukes til å overføre en (el. flere) verdier fra prosedyre/funksjons kallet. På den måten kan prosedyren el. funksjonen gjøres mer generell. La oss se på funksjonen *Formatter* som vi brukte i eksempelet ovenfor. Funksjonen har en parameter – Verdi – med datatypen "Single". Foran står det reserverte ordet *ByVal*, som betyr at vi kopierer verdien fra funksjonskallet til selve funksjonen. Siden funksjonen kun arbeider med en kopi vil ikke verdien til den originale variabelen kunne endres.

Men i noen tilfeller ønsker vi å endre verdien til prosedyre– el. funksjonskallet, da sløyfer vi det reserverte ordet *ByVal*, dermed vil prosedyren el. funksjonen arbeide med den originale variabelen. Og eventuelle endringer som gjøres i prosedyren el. funksjonen blir reflektert i funksjonskallets variabel.

La oss se på et enkelt eksempel med to prosedyrer. Den ene deklarerer med *ByVal* foran parameteren, den andre ikke. Start et nytt VB prosjekt, og legg til en tekstboks, en trykknapp og to etiketter på formvinduet. Dobbeltklikk på trykknappen og legg inn denne koden:

```
Private Sub Command1_Click()
  Dim x As Integer, y As Integer
  x = Val(Text1.Text)      ' La x få verdien bruker skriver inn
  y = x                    ' La y få samme verdi som x
  ByValProsedyre x        ' Kall ByValProsedyre
  ByRefProsedyre y        ' Kall ByRefProsedyre
  Label1.Caption = "ByVal: " & Str$(x)
  Label2.Caption = "ByRef: " & Str$(y)
End Sub
```

Legg så til en modul og skriv inn disse to prosedyrene:

```
Public Sub ByValProsedyre(ByVal i As Integer)
  i = 32530
End Sub

Public Sub ByRefProsedyre(i As Integer)
  i = 32530
End Sub
```

Lagre programmet og kjør det. Tast inn en liten verdi i tekstboksen og klikk på trykknappen. Hvilken prosedyre endrer verdi?

Filbehandling

En fil er en samling data som lagres på ytre lager (diskett/harddisk). Dermed kan informasjon lagres slik at en slipper å taste inn den samme informasjonen hver gang programmet kjøres. Dersom datamengden er stor må den lagres på ytre lager fordi den interne hukommelsen ikke er stor nok.

Det er to hovedtyper filer:

- **Tekstfiler** er en fortløpende strøm av tegn. Linjene kan ha ulik lengde, et linjesluttmerke (ASCII kodene 13 og 10) skiller en linje fra den neste. Tekstfilene er sekvensielle, dvs. de må leses fra begynnelse til slutt.
- **Direkte adgang filer** (postbaserte filer/"Random Access Files") er organisert i poster. Hver post har samme størrelse, dette gjør at filsystemet kan hoppe direkte til en bestemt post og endre den.

Tekstfiler

Tekstfiler åpnes enten for skriving til fila, eller for lesing fra fila.

For å åpne en fil for skriving til fila (dersom fila finnes fra før, slettes den gamle, ellers opprettes ny fil):

```
Open filnavn For Output As #filnummer
```

For å åpne en fil for lesing fra fila (fila må finnes fra før):

```
Open filnavn For Input As #filnummer
```

Filnummer er en heltallsvariabel (som #1, #2, #3 etc).

For å skrive en linje til en tekstfil (som er åpnet med *Open ... For Output ...*) brukes setningen:

```
Print #filnummer, variabel
```

Der "variabel" har datatypen *String*.

For å lese en linje fra en tekstfil (som er åpnet med *Open ... For Input ...*) brukes setningen:

```
Input #filnummer, variabel
```

Også her er "variabel" av datatypen *String*.

En fil må alltid lukkes nå du er ferdig med å bruke den. Det gjøres med kommandoen:

```
Close #filnummer
```

Her kommer et lite eksempel som viser hvordan du leser en tekstfil (kalt TEST.TXT) og skriver innholdet ut på formvinduet:

```
Dim Linje As String
Open "H:\VB\TEST.TXT" For Input As #1
While Not EOF(1)
    Input #1, Linje
    Print Linje
Wend
Close #1
```

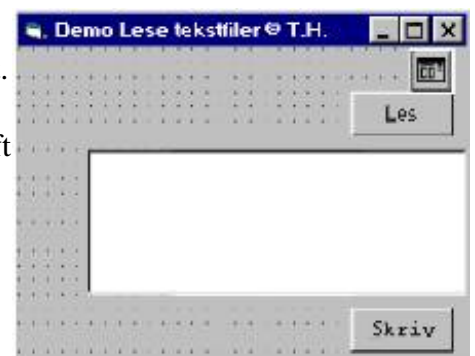
Bruke "Åpne" dilaogen til å hente filnavn

For å kunne bruke Windows standard *Åpne* dialog må den være installert. *Åpne* er en del av Windows "Common Dialog Control". Dersom VB 5 CCE mangler denne kontrollen, velg **Project, Components**. Finn "Microsoft Common Dialog Control" og kryss av den og klikk knappene "Bruk" og "OK".

La oss bruke "Åpne" kontrollen i lag med en tekstboks til å lage et enkelt tekstredigeringsverktøy. Start et nytt VB prosjekt. Plasser en "Åpne" kontroll i Form vinduet, og gi den navnet *CmDialog1*. Legg inn et par kommandoknapper og en tekstboks (se fig. til høyre).

Dobbelklikk på "Les" knappen og legg inn denne koden:

```
Private Sub CmdLes_Click()
    Dim FilNavn As String, Linje As String
    CmDialog1.Filter = "Tekstfiler (*.txt)|*.txt|Alle filer (*.*)|*.*"
    CmDialog1.FilterIndex = 1
    CmDialog1.Action = 1          ' Vis Åpne dialog.
```



Introduksjon til Visual Basic 5 (TH)

```
FilNavn = CmDialog1.filename ' Hent valgt filnavn, Cancel gir tom
If FilNavn <> "" Then
  Cls
  Open FilNavn For Input As #1 ' Åpne fil for lesing fra den
  While Not EOF(1)
    Input #1, Linje           ' Les en linje
    ' Legg linja inn i tekstboksen
    Text1.Text = Text1.Text &Linje &Chr(13) &Chr(10)
  Wend
  Close #1                   ' Lukk fila
End If
End Sub
```

Filter egenskapen til Åpne dialogen angir hvilke filtyper som kan velges, *FilterIndex = 1* angir at første valgmulighet (Tekstfiler) skal være aktiv. *Action = 1* angir at vi ønsker Åpne dialogen (2 vil gi Lagre dialogen).

Tekstboksen sin egenskap *MultiLine* må være satt til "True". Linjen *Text1.Text = Text1.Text &Linje &Chr(13) &Chr(10)* legger hver linje inn i tekstboksen. *Chr(13)* og *Chr(10)* brukes til å lage ny linje i tekstboksen.

Vi kan nå lese en liten tekstfil inn i redigeringsverktøyet vårt. Neste steg blir å lagre eventuelle endringer. Koden for "Skriv" knappen blir slik:

```
Private Sub CmdSkriv_Click()
  Dim FilNavn As String, Linje As String
  Linje = Text1.Text
  CmDialog1.Filter = "Tekstfiler (*.txt)|*.txt|Alle filer (*.*)|*.*"
  CmDialog1.FilterIndex = 1
  CmDialog1.ShowSave           ' Vis Lagre dialog.
  FilNavn = CmDialog1.filename ' Hent valgt filnavn
  If FilNavn <> "" Then
    Open FilNavn For Output As #1 ' Åpne fil for skriving til den
    Print #1, Linje              ' Skriv tekst til fila
    Close #1
  End If
  Text1.Text = ""              ' Slett lagret tekst ??
End Sub
```

Her bruker vi *CmDialog.ShowSave* i stedet for *Action* (*ShowOpen* alternativet viser Åpne dialogen). Etter at fila er lagret slettes innholdet i tekstboksen, dette burde muligens heller gjøres i starten av koden bak "Les" knappen.

Direkte adgang filer

For å kunne bruke direkte adgang filer må vi først definere en post. Dette bør gjøres i en egen modul (velg Project, Add Module). Dermed kan postdefinisjonen gjøres kjent for alle moduler i programmet med det reserverte ordet *Public*:

```
Public Type MinPost
  Navn As String *20
  Alder As Integer
End Type
```

Denne postdefinisjonen kan så brukes av alle deler av programmet. Legg inn to tekstbokser i formvinduet. Legg så til to trykknapper på formvinduet (sett "Caption" til den første til *Lagre post*, den andre til *Vis post*). Her kommer koden til *Lagre post*:

Introduksjon til Visual Basic 5 (TH)

```
Dim Post As MinPost
Open "POSTEST.DAT" For Random As #1 Len = Len(Post) ' Åpne(el. opprett) datafila
Post.Navn = Text1.Text                               ' Fyll postens felter
Post.Alder = Val(Text2.Text)
Put #1, 1, Post                                     ' Lagre postdata
Close #1                                             ' Lukk datafila
```

Koden for *Vis post* knappen blir slik:

```
Dim Post As MinPost
Open "POSTEST.DAT" For Random As #1 Len = Len(Post) ' Åpne datafila
Get #1, 1, Post                                     ' Les post
Print "Navn: "; Post.Navn; ", alder: "; Str$(Post.Alder)
Close #1
```

Problemet med dette enkle eksemplet er at vi bare kan lagre (el. lese) en post (nummer 1). I et virkelig program må det være mulig å føye til nye poster bakerst i fila. Det vil si at i linja *Put #1, 1, Post* så må tallet 1 endres til 2 når vi skriver den andre posten. 3 når vi skriver den tredje osv. Det vil si vi må ha en variabel – *PostNr* – med datatype *Long*.

La oss nå se på et eksempel med tre vinduer, det første vinduet utgjør menyen, vindu 2 er inndatavinduet, og vindu 3 viser innholdet i datafila slik at vi kan bla oss fra post til post.

Start et nytt VB prosjekt og legg til tre trykknapper. For den første knappen settes egenskapen "Caption" til *Ny post*, den andre til *Vis poster* og den siste til *Avslutt*. Legg inn passende kode for hva de tre knappene skal utføre.

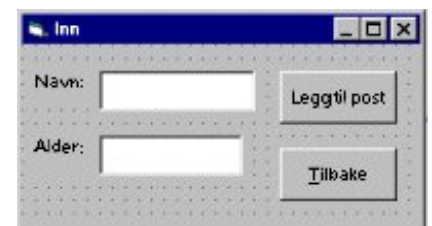
Legg så til en modulfil, og legg inn denne koden:

```
Public Type MinPost
    Navn As String *20
    Alder As Integer
End Type

Public Const FilNavn = "H:\TEST.DAT"
```

Legg så til et nytt formvindu (Form2) med et par tekstbokser og et par trykknapper, se fig. til høyre:

Form_Activate og trykknappene skal ha denne koden:



```
Dim PostNr As Long

Private Sub CmdLeggTil_Click()
    Dim Post As MinPost
    Post.Navn = Text1.Text           ' Kopier data til post
    Post.Alder = Val(Text2.Text)
    Put #1, PostNr, Post             ' Skriv post til fil
    PostNr = PostNr + 1             ' Pek på neste post
End Sub

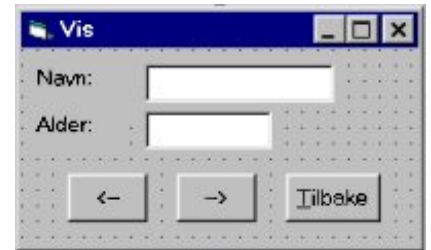
Private Sub Form_Activate()
    Dim Post As MinPost
    If Dir(FilNavn) <> "" Then
        Open FilNavn For Random As #1 Len = Len(Post)
        PostNr = LOF(1) \ Len(Post) + 1
    Else
```

Introduksjon til Visual Basic 5 (TH)

```
MsgBox "Datafila opprettes"  
PostNr = 1  
Open FilNavn For Random As #1 Len = Len(Post)  
End If  
End Sub  
  
Private Sub CmdTilbake_Click()  
Close #1  
Form2.Hide  
Form1.Show  
End Sub
```

Legg merke til at variabelen PostNr er definert utenfor alle prosedyrene, på denne måten blir denne variabelen tilgjengelig for alle prosedyrene i dette formvinduet. Og variabelen opprettes når vinduet opprettes, og den "forsvinner" ikke før programmet avsluttes.

Opprett så et tredje formvindu (Form3), slik:
Og legg inn denne koden:



```
Dim PostNr As Long  
  
Private Sub Form_Activate()  
Dim Post As MinPost  
If Dir(FilNavn) <> "" Then  
Open FilNavn For Random As #1 Len = Len(Post)  
PostNr = 1  
Get #1, PostNr, Post  
Text1.Text = Post.Navn  
Text2.Text = Post.Alder  
Else  
MsgBox "Datafila må opprettes først!"  
CmdTilbake_Click  
End If  
End Sub  
  
Private Sub CmdForrige_Click()  
Dim Post As MinPost  
If PostNr > 1 Then  
PostNr = PostNr - 1 ' Gå til forrige post  
Get #1, PostNr, Post  
Text1.Text = Post.Navn  
Text2.Text = Post.Alder  
End If  
End Sub  
  
Private Sub CmdNeste_Click()  
Dim Post As MinPost  
If PostNr < LOF(1) \ Len(Post) Then  
PostNr = PostNr + 1 ' Gå til neste post  
Get #1, PostNr, Post  
Text1.Text = Post.Navn  
Text2.Text = Post.Alder  
End If  
End Sub  
  
Private Sub CmdTilbake_Click()  
Close #1 ' Husk å lukke fila når vi forlater dette vinduet.  
Form3.Hide
```

```
Form1.Show
End Sub
```

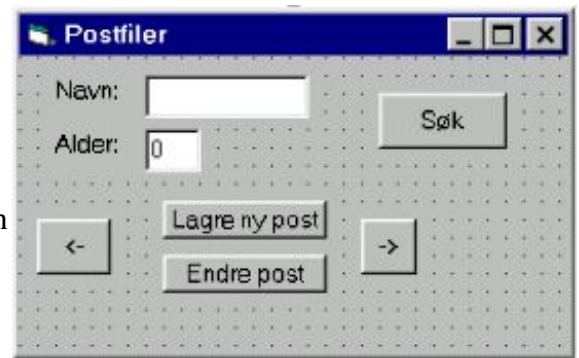
Dersom du ønsker å kunne endre innholdet i en post fra dette vinduet, plasserer du en kommandoknapp til over "Tilbake" knappen. Sett egenskapen "Caption" til *Endre*. Legg inn denne koden for *Endre* knappen:

```
Private Sub CmdEndre_Click()
    Dim Post As MinPost
    Post.Navn = Text1.Text           ' Kopier data til post
    Post.Alder = Val(Text2.Text)
    Put #1, PostNr, Post           ' Skriv post til fil
End Sub
```

Et mer avansert redigeringsvindu

Dersom det er mange poster i datafila vil brukeren måtte bla mye fram og tilbake for å finne en bestemt post, en søkerutine vil løse dette problemet. Det ville også være praktisk å kunne legge til en ny post, og endre innholdet i en eksisterende post fra samme formvindu. La oss endre eksemplet vårt slik at et vindu håndterer både innlegging av nye poster, endring av poster og søking etter en post. Sålenge filene ikke er sorterte må vi bruke en sekvensiell søkerutine som leser gjennom hver post til ønsket post eventuelt finnes.

Lag et nytt formvindu som vist til høyre, VB koden til vinduet er slik:



```
Dim AntPoster As Long, PostNr As Long

Private Sub Vis(Post As tMinPost)
    Text1.Text = Post.Navn         ' Vis post i tekstbokser
    Text2.Text = Post.Alder
End Sub

Private Sub Form_Activate()
    Dim Post As tMinPost
    If Dir(FilNavn) = "" Then      ' Fila finnes ikke fra før.
        Open FilNavn For Random As #1 Len = Len(Post)
        AntPoster = 0              ' Derfor ingen poster i fila
        PostNr = 0
        MsgBox "Oppretter datafila."
    Else                            ' Fila finnes fra før
        Open FilNavn For Random As #1 Len = Len(Post)
        If LOF(1) > 0 Then
            AntPoster = LOF(1) \ Len(Post)
            PostNr = 1
            Get #1, PostNr, Post
            Vis Post
        Else
            MsgBox "Datafila er tom!"
            AntPoster = 0
            PostNr = 0
        End If
    End If
End Sub
```

Introduksjon til Visual Basic 5 (TH)

```
End If
End Sub

Private Sub CmdLagreNyPost_Click()
    Dim Post As tMinPost
    AntPoster = LOF(1) \ Len(Post)
    PostNr = AntPoster + 1           ' Legg til ny bak siste post
    Post.Navn = Text1.Text
    Post.Alder = Val(Text2.Text)
    Put #1, PostNr, Post
    AntPoster = LOF(1) \ Len(Post)
End Sub

Private Sub CmdEndrePost_Click()
    Dim Post As tMinPost
    Post.Navn = Text1.Text
    Post.Alder = Val(Text2.Text)
    Put #1, PostNr, Post
End Sub

Private Sub CmdForrige_Click()
    Dim Post As tMinPost
    If PostNr > 1 Then
        PostNr = PostNr - 1
        Get #1, PostNr, Post
        Vis Post
    End If
End Sub

Private Sub CmdNeste_Click()
    Dim Post As tMinPost
    AntPoster = LOF(1) \ Len(Post)
    If PostNr < AntPoster Then
        PostNr = PostNr + 1
        Get #1, PostNr, Post
        Vis Post
    End If
End Sub

Private Sub CmdSøk_Click()
    Dim P As tMinPost
    P.Navn = InputBox("Tast navnet du leter etter")
    If Finn(P) > 0 Then ' Post funnet
        Text1.Text = P.Navn
        Text2.Text = P.Alder
    Else
        MsgBox "Ikke funnet! " & P.Navn
        Text1.Text = ""
        Text2.Text = "0"
    End If
End Sub
```

VB koden for søkerutinen plasserer vi i modulfilen, som blir slik:

```
Public Type tMinPost
    Navn As String * 20
    Alder As Integer
End Type
```

Introduksjon til Visual Basic 5 (TH)

```
Public Const FilNavn = "H:\TEST.DAT"

Public Function Finn(Post As tMinPost) As Long
    Dim AntPoster As Long, PostNr As Long
    Dim P As tMinPost, Funnet As Boolean, FilNr As Integer
    PostNr = 1 ' Start med første post
    Funnet = False ' Posten ikke funnet
    FilNr = FreeFile(255) ' Finn ledig filnummer
    Open FilNavn For Random As #FilNr Len = Len(Post)
    If LOF(FilNr) < 1 Then ' Ingen poster i fila,
        Finn = -1 ' signaliser feil
        Post.Navn = "Datafil tom." ' og avbryt.
        Close #FilNr
        Exit Function
    End If
    AntPoster = LOF(FilNr) \ Len(Post)
    While (Not Funnet) And (PostNr <= AntPoster)
        Get #FilNr, PostNr, P
        If UCase(P.Navn) = UCase(Post.Navn) Then
            Funnet = True
        Else
            PostNr = PostNr + 1
        End If
    Wend
    Close #FilNr
    If Funnet Then
        Post = P ' Returner funnet post
        Finn = PostNr ' Returner PostNr til funnet post
    Else
        Post.Navn = "Ikke funnet"
        Finn = -1 ' -1 signaliserer ikke funnet
    End If
End Function
```

I søkerutinen "Finn" brukes funksjonen *FreeFile* som returnerer første ledige filnummer. Modulfilen vet ikke hvilke filnummer som allerede er brukt, derfor er det lurt å la VB finne et ledig filnummer for oss. Parameteren Post brukes til å returnere postinnholdet dersom søkt post blir funnet. Ellers vil parameteren Post returnere teksten "Ikke funnet".

Inndatakontroll

Et ordtak innen databehandling er slik: "Søppel inn gir søppel ut!". Dersom de data som programmet får inn er feil, er det ikke mulig for programmet å gi fornuftige utdata!

Programmene vi lager bør derfor prøve å kontrollere om den informasjon som brukerne taster inn er fornuftig. La oss gå tilbake til programeksemplet for postkontoret (s.11). Programmet skulle brukes for å få oversikt over portoene kundene må betale for pakkepost. Dess tyngre pakke dess høyere porto. Programmet bruker et tekstfelt for inntasting av pakkens vekt. En negativ inntasting vil være meningsløs, og programmet bør derfor gi en feilmelding dersom vekten er mindre enn 0. Koden vi brukte var slik:

```
Private Sub CmdSummer_Click()
    Dim Porto As Single, Vekt As Single, Ant As Integer
    Vekt = Val(Text1.Text)
    Ant = Val(Text2.Text)
    If Vekt < 1 Then
        Porto = 15 * Ant
        AntUlkkg = AntUlkkg + 1
    End If
End Sub
```

Introduksjon til Visual Basic 5 (TH)

```
Else
    Porto = 45 * Ant
    AntOlkkg = AntOlkkg + 1
End If
Sum = Sum + Porto
LblSum.Caption = Format$(Sum, "0.00")
SluttSum = SluttSum + Porto
End Sub
```

For å kontrollere at variabelen *Vekt* er større enn null legger vi til en løkke etter tredje linje. Vi bruker en *If* setning til å sjekke om *Vekt* er mindre enn el. lik null. Dersom det er tilfelle vises en feilmelding, og vi ber brukeren taste inn vekten på nytt. Løkkja avsluttes ikke før *Vekt* er større enn null:

```
Private Sub CmdSummer_Click()
    Dim Porto As Single, Vekt As Single, Ant As Integer
    Vekt = Val(Text1.Text)
    Do
        If Vekt <= 0 Then
            ' Hvis Vekt er negativ el. 0
            MsgBox ("NB. Vekt må være større enn null!")
            Vekt = Val(InputBox("Tast vekt")) ' Les inn Vekt på nytt
            Text1.Text = Vekt
            ' Oppdater tekstfeltet
        End If
        Loop Until Vekt > 0
        Ant = Val(Text2.Text)
        If Vekt < 1 Then
            Porto = 15 * Ant
            AntUlkkg = AntUlkkg + 1
        Else
            Porto = 45 * Ant
            AntOlkkg = AntOlkkg + 1
        End If
        Sum = Sum + Porto
        LblSum.Caption = Format$(Sum, "0.00")
        SluttSum = SluttSum + Porto
    End Sub
```

Prøv nå å gjøre det samme med variabelen *Ant* slik at kun verdier større enn null blir godtatt. Dersom det er en dato som tastes inn kan du bruke funksjonen *IsDate(parameter)* for å kontrollere om parameter er en gyldig dato. Funksjonen *Now()* returnerer dagens dato og klokkeslett.

Den inndatakontrollen som er vist i eksemplet over, vil ikke håndtere inntasting av f.eks. bokstaver i innfeltet for vekt. Dersom vi ønsker å forhindre at uvedkommende tegn tastes inn i tekstboksen, kan vi bruke *KeyPress* hendelsen. Dobbelklikk i tekstboksen for inntasting av vekt, velg *KeyPress* isteden for *Change* og legg inn denne koden:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    Select Case KeyAscii
        Case vbKey0 To vbKey9
            ' Vi ønsker disse, så ikke gjør noe
        Case Is = vbKeyBack
            ' Vi ønsker å kunne slette, så ikke gjør noe med 'Backspace'
        Case Asc("-")
            MsgBox ("NB. Bare positive tall!")
            KeyAscii = 0
        Case Asc(".")
            If InStr(Text1, ".") Then
                MsgBox ("NB. Bare et desimaltegn!")
                KeyAscii = 0
            End If
        Case Else
            ' Vi ønsker ingen andre tegn
            KeyAscii = 0
    End Select
```

Introduksjon til Visual Basic 5 (TH)

```
End If
Case Else ' Ønsker ikke resten, kasser dem
  MsgBox ("NB. Feil inndata!" & Chr(10) & _
    "Tast kun tall el. desimalpunktum!")
  KeyAscii = 0
End Select
End Sub
```

Her blir talltastene 0 til 9 (vbKey0 To vbKey9) akseptert. Ett desimaltegn (punktum) blir akseptert, i tillegg tillater vi at "Backspace" tasten brukes. Ellers vil ikke bruker kunne slette tegn.

Tabeller (indekserte variabler)

Til nå har vi arbeidet med variabler med ulike navn. Gå tilbake til oppgaven å finne det største av to inntastede tall. Der er det naturlig å bruke to variabler – *Tall1* og *Tall2*, og så bruke en *If* setning til å finne ut hvilket tall som er størst. Koden kan være omtrent slik:

```
Dim Tall1 As Integer, Tall2 As Integer
Tall1 = Val(Text1.Text)
Tall2 = Val(Text2.Text)
If Tall1 > Tall2 Then
  MsgBox "Første tall er størst"
Else
  MsgBox "Andre tall er størst"
End If
```

Men dersom oppgaven endres slik at vi skal finne det største av tre tall? Vi kunne deklare en ny variabel – *Tall3* – og bruke flere *If* setninger. Dersom vi ønsker å finne det største (og det minste) av flere tall, så blir en slik fremgangsmåte omstendelig.

Det ville være mye enklere om vi kunne bruke en variabel med plass til alle de ulike tallene. Her er det *tabeller* kommer inn. En tabell er en variabel som inneholder en rekke elementer av samme datatype. Vi får adgang til et element ved hjelp av en indeks. La oss se på et eksempel:

```
Dim Tabell(1 To 5) As Integer ' Deklarer tabell
Private Sub Command1_Click()
  Dim i As Integer ' Deklarer indeksvariabel
  For i = 1 To 5
    Tabell(i) = Val(Inputbox("Tast et heltall"))
  Next i
End Sub
```

I den første linjen opprettes en tabell med fem elementer. Det første elementet har indeks en, det andre indeks to osv.

Vi legger en verdi inn i et element i tabellen med: *Tabell(i) = Val(Inputbox("Tast et heltall"))*.

La oss se på et prosjekt som leser inn fem tall, og sorterer tallene i stigende rekkefølge. I eksemplet ovenfor deklarerer vi tabell utenfor prosedyren. En tabell kan ikke deklarerer med *Dim* inni en prosedyre. Det kan være lurt å deklare tabeller i en egen modulfil som *Global*. Start derfor et nytt prosjekt, sett egenskapen *Autoredraw* til True. Legg til en modulfil (Project, Add Module) og legg inn denne koden i modulfilen:

```
Public Const N = 5
Global TallTabell(1 To N) As Integer

Public Sub Bytt(T1 As Integer, T2 As Integer)
```

Introduksjon til Visual Basic 5 (TH)

```
Dim Tmp As Integer
Tmp = T1
T1 = T2
T2 = Tmp
End Sub

Public Sub Sorter()
Dim i As Integer, Sortert As Boolean
Do
Sortert = True
For i = 2 To N
If TallTabell(i) < TallTabell(i - 1) Then
Bytt TallTabell(i), TallTabell(i - 1)
Sortert = False
Next i
Loop Until Sortert
End Sub
```

Her deklarerer tabellen med det reserverte ordet *Global* slik at tabellen blir synlig for alle deler av prosjektet. Konstanten *N* brukes til å angi antall elementer i tabellen. Den er definert med det reserverte ordet *Public* slik at den er synlig for alle deler av prosjektet. Dersom vi senere ønsker en tabell med flere elementer, behøver vi kun endre verdien til denne konstanten.

Prosedyren *Bytt* bytter om verdien til de to parametrene *T1* og *T2*, og returnerer dem til den kallende prosedyren.

Prosedyren *Sorter* vil sortere tabellen i stigende rekkefølge slik at det minste tallet plasseres i første element, det største tallet i det siste elementet.

Legg til en kommandoknapp i formvinduet. Koden til kommandoknappen er slik:

```
Private Sub Command1_Click()
Dim i As Integer;
For i = 1 To N          ' Les verdier inn i tabellen
TallTabell(i) = Val(Inputbox("Tast et heltall"))
Next i
Print "Usortert"
For i = 1 To N
Print TallTabell(i); ", "
Next i
Print
Sorter                  ' Kall rutine som sorterer tabellen
Print "Sortert"
For i = 1 To N
Print TallTabell(i); ", "
Next i
End Sub
```

Du kan deklare en tabell (utenfor alle prosedyrer) uten å angi størrelsen på tabellen:

```
Dim Tab() As Integer
```

Senere kan du inni en prosedyre endre antall elementer med

```
Redim Tab(5)
```

Med *Redim* er det mulig å lage dynamiske tabeller, dvs. du kan endre størrelsen på tabellen mens programmet kjører.

Dersom du allerede har lagt informasjon inn i tabellen, kan du bruke

Redim Preserve Tab(8)

Du kan selvsagt også bytte ut tallet 8 i parentesen med en heltallsvariabel. Dermed kan du la den som bruker programmet bestemme hvor stor tabellen *Tab* skal bli.

Visual Basic konstanter

Visual Basic har en rekke predefinerte konstanter (se f.eks. Mer om meldingsboksen, s. 5 – 7). Her kommer noen flere tegnkonstanter og fargekonstanter:

vbCrLf	Chr(13) + Chr(10)	Linjesluttmerke (CrLf)
vbCr	Chr(13)	Vognretur (CR)
vbLf	Chr(10)	Ny linje (LF)
vbNullString		Tom streng ("")
vbTab	Chr(9)	Tabulator (TAB)
vbBack	Chr(8)	Slett ← (BackSpace)
vbBlack	0x0	Svart
vbRed	0xFF0	Rød
vbGreen	0xFF00	Grønn
vbYellow	0xFFFF	Gul
vbBlue	0xFF0000	Blå
vbMagenta	0xFF00FF	Magenta
vbCyan	0xFFFF00	Cyan
vbWhite	0xFFFFFFFF	Hvit