Sisällys

Lukijalle 1
1. Yleistä....3
1.1 Käsitteiden määrittelyä 3
1.2 Ohjelmointiympäristö 4
1.3 Esimerkkiohjelman luominen 4
1.3.1 Käyttöliittymän piirtäminen 6
1.3.2 Ominaisuuksien arvojen asettaminen 6
1.3.3 Koodin kirjoittaminen 8
2. Lomakkeet ja muuttujat....13
2.1 Lomakkeet....13
2.2 Muuttujat, datatyypit ja vakiot 13
2.2.2 Tietueet 16
2.2.3 Vakiot....17
3. Ohjaimista 19
3.1 Otsikko (Label) 19
3.2 Tekstilaatikko (TextBox)....20
3.3 Yhdistelmälista (ComboBox) 20
3.4 Lista (ListBox) 21
3.5 Tiedostolistat (DriveListBox, DirectoryListBox, FileListBox) 22
3.6 Kuvakehys ja Kuva (PictureBox, Image)....24
3.7 Valintaruudut ja -napit (CheckBox, OptionButton)....25
3.8 Vierityspalkit (ScrollBar)....26
3.9 Ajastin (Timer) 27
3.10 Ominaisuuksien arvot ja niiden muuttaminen (esimerkkinä nappi) 28
4. Tapahtumia....29
4.1 Hiiritapahtumia 29
4.1.1 MouseUp ja MouseDown 29
4.1.2 MouseMove 29
4.1.3 Drag And Drop 29
4.2 Näppäintapahtumia 31
4.2.1 KeyUp ja KeyDown....31
4.2.2 KeyPress 32
4.3 Paint-tapahtuma (kuvakehys, lomake)....32
5. Ohjelmointikieli 35
5.1 Kontrollirakenteet 35
5.1.1 If....35
5.1.2 Select Case....35
5.1.3 Do...Loop 36
5.1.4 For...Next 36
5.2 Taulukot....36
5.2.1 Yksiulotteiset taulukot 36
5.2.2 Ohjaintaulukko....37
5.2.3 Matriisit....38
5.2.4 Dynaamiset taulukot 38
5.3 Aliohjelmat 39
5.3.1 Sub 40

5.3.2 Function 40
5.3.3 ByVal 41
6. Debuggaus 43
7. Kehittyneemmät ohjelmat 45
7.1 Useamman lomakkeen ohjelmat 45
7.2 Globaalit Forms- ja Controls-taulukot 46
7.2.1 Forms-taulukko 46
7.2.2 Controls-taulukko 46
7.3 Lomakkeen esiintymät 46
7.4 Valikot (Menu) 47
7.5 Olio-osoittimet 49
7.6 Olio-osoitintaulukot 50
7.7 Taulukko-ohjain (Grid) 51
7.8 Vakiodialogit 52
7.8.1 Tiedoston avaus (Open) 52
7.8.2 Tiedoston talletus nimellä (Save As) 53
7.8.3 Värin vaihto (Color) 54
7.8.4 Fontin vaihto (Font) 54
7.8.5 Tulostus (Print) 55
7.9 Viestidialogi (MessageBox) 55
7.10 Syöttödialogi (InputBox) 56
7.11 Virheenkäsittely 57
7.12 Avustustiedostot 58
7.13 MDI (Multiple-Document Interface) 58
7.14 DLL (Dynamic-link libraries) 60
7.14.1 Valmiiden funktioiden kutsuminen 61
7.14.2 Itsetehtyjen aliohjelmien kutsuminen 61
8. Tiedonsiirto muiden ohjelmien kanssa 65
8.1 Tiedostot 65
8.1.1 Suorasaantitiedostot (Random) 65
8.1.2 Peräkkäistiedostot (Sequential) 66
8.1.3 Binääritiedostot (Binary) 67
8.2 Leikepöytä 67
8.2.1 Tekstin siirtäminen 68
8.2.2 Kuvien siirtäminen 69
8.3 DDE (Dynamic Data Exchange) 69
8.3.1 Linkit ilman koodia (Paste Link) 69
8.3.2 Linkit koodista 70
8.3.3 Ohjelmaan Liitä linkki -valikkokomento (Paste Link) 71
9. Tietokanta 73
Kirjallisuutta 77
Liite 1. Ohjaimet ja niiden yleisimpiä ominaisuuksia 79
Liite 2. Erikoisoliot 84
Liite 3. Lisää ominaisuuksia 86
Liite 4. Tapahtumia 87
Liite 5. Metodeja 89
Liite 6. Ohjainten oletusominaisuudet 92
Liite 7. Ongelman ratkaisu 1. esimerkkiin 93
Hakemisto 95

Lukijalle

Tämän monisteen tarkoituksena on antaa perustietoja Visual Basic 3.0 -ohjelmointikielestä ja auttaa pääsemään ohjelmoinnin alkuun. Monisteen tekijä on itsekin vasta monistetta tehdessään perehtynyt kyseiseen ohjelmointikieleen.

Moniste on tehty oppilastyönä Jyväskylän yliopistossa.

Monisteessa mainitut esimerkkiohjelmat on saatavissa levykkeellä.

Kritiikki, rakentava palaute ja parannusehdotukset otetaan mielihyvin vastaan. Niitä voi lähettää esimerkiksi sähköpostina osoitteella kaarina@jyu.fi.

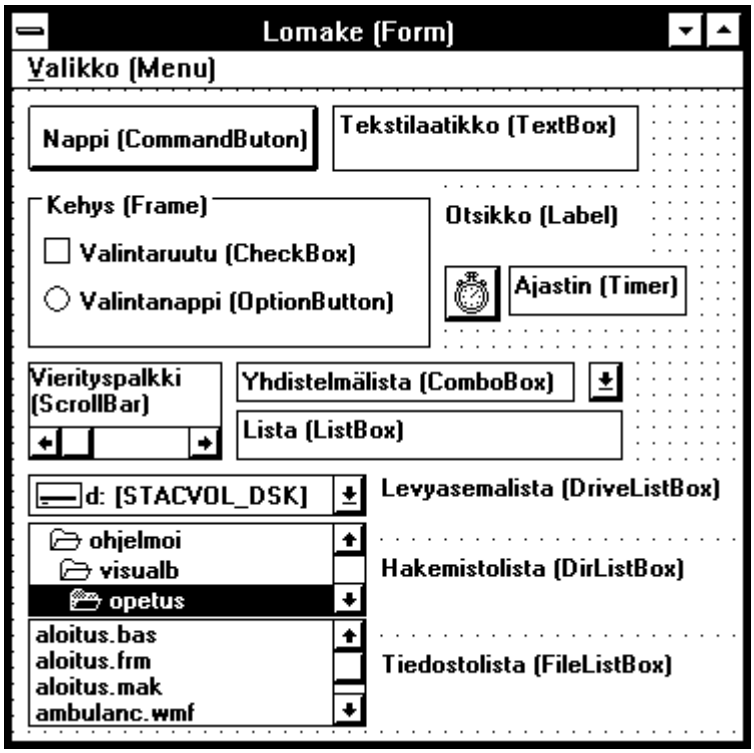
Jyväskylässä lokakuussa 1994

Kaarina Mäenpää

1. Yleistä

1.1 Käsitteiden määrittelyä

Alla olevassa kuvassa ja taulukossa on esitelty tässä monisteessa käytettyjä termien suomennoksia.



Lomakkeen olioita kutsutaan ohjaimiksi (**Control**). Ohjainten ominaisuuksille (**Properties**) asetetaan arvot. Ohjaimiin liittyy tapahtumia (**Events**) sekä metodeja (**Methods**) esim. graafiset metodit ks. liite 5.

ajastin	Timer	
esiintymä	Instance	
hakemistolista	Directory List Box	
kehys	Frame	
koodi-ikkuna	Code Window	
kuva	Image	
kuvakehys	Picture Box	
levyasemalista	Drive List Box	
lista	List Box	
lomake	Form	
metodi	Method	
nappi	Command Button	Push button
ohjain	Control	
olio	Object	
ominaisuus	Property	

ominaisuusikkuna	Properties Window	
otsikko	Label	Static text
projekti-ikkuna	Project Window	
tapahtuma	Event	
tekstilaatikko	Text Box	Edit text
tiedostolista	File List Box	
työkalulaatikko	Toolbox	
valikko	Menu	
valintanappi	Option Button	Radio button
valintaruutu	Check Box	
vierityspalkki	Scrollbar	
yhdistelmälista	Combo Box	

1.2 Ohjelmointiympäristö

Visual Basic on ohjelmointiympäristö, jossa Windows-ohjelmien rakentaminen on pyritty tekemään mahdollisimman helpoksi. Ohjelman suunnittelu alkaa käyttöliittymän piirtämisellä. Tämän jälkeen Windows-olioille (esim. napeille) asetetaan ominaisuuksien arvot (ominaisuuksia ovat esim. taustäväri, fontin koko jne.) ja tapahtumille halutut aliohjelmat. Tyypillinen tapahtuma on jokin käyttäjän ajon aikana suorittama hiiri- tai näppäintoiminto (esim. napin klikkaaminen hiirellä, josta tulee Click-tapahtuma). Se voi olla myös jokin järjestelmän sisäinen tapahtuma, esim. jos tekstilaatikon teksti muuttuu, siitä tulee aina automaattisesti Change-tapahtuma.

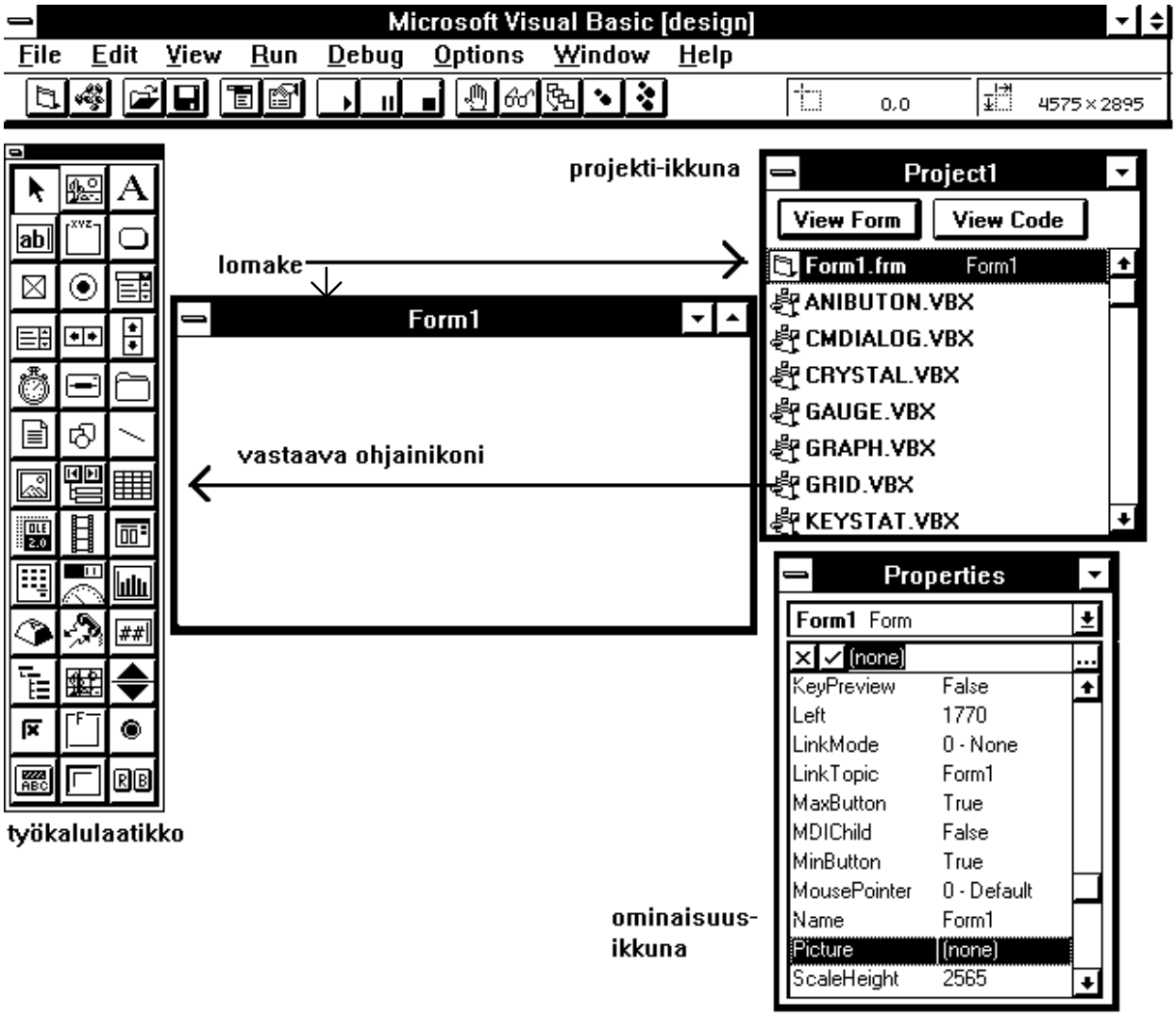
Tapahtumaohjattu ohjelma koostuu olioista, jotka ovat passiivisia, kunnes niihin kohdistuu jokin tapahtuma. Tällöin olio saa viestin, että sen kohdalla on esimerkiksi painettu hiiren nappia. Ohjelma reagoi tähän viestiin toteuttamalla tapahtumaan liitetyn aliohjelman. VB:ssa oliot ovat joko työkalulaatikosta valittuja ohjaimia, valikkoja tai lomakkeita. Lisäksi VB tuntee viisi erikoisoliota: sovellus, leikepöytä, näyttö, tulostin ja virheenetsintä (ks. Liite 2). Erikoisolioita ei tarvitse erikseen lisätä ohjelmaan vaan ne ovat automaattisesti käytettävissä joka ohjelmassa. Erikoisolioita voi esiintyä vain yhdet ohjelmaa kohti.

Visual Basic -ohjelma koostuu yhdestä tai useammasta lomakkeesta, jotka näkyvät käyttäjälle erillisinä ikkunoina. Kaikki lomakkeet talletetaan erillisiksi tiedostoiksi, joihin tallentuu lomakkeiden ja ohjainasetusten lisäksi niihin liitetty koodi. Lomakkeiden lisäksi koodia voidaan kirjoittaa myös erillisiin moduulitiedostoihin. Näissä tiedostoissa on yhteisiä aliohjelmia, joita voidaan kutsutaan esim. lomakkeen ohjainaliohjelmista.

1.3 Esimerkkiohjelman luominen

Kun VB käynnistetään, ruudulle ilmestyy neljä ikkunaa: (ongelmanratkaisuesimerkki ks. liite 7.)

- työkalulaatikko,
- lomake,
- projekti-ikkuna ja
- ominaisuusikkuna.

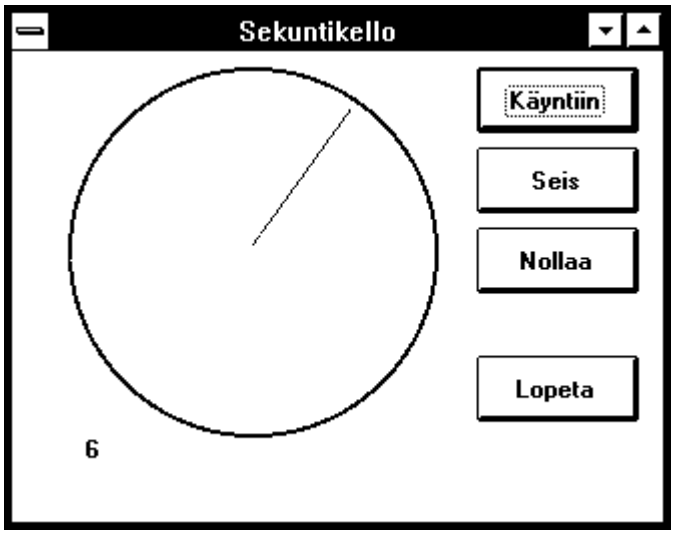


Projekti sisältää jokaisen ohjelmaan liitetyn lomakkeen (**.FRM**), koodimoduuleita (**.BAS**) ja **.VBX**-tiedostoja, jotka sisältävät tiedon asennetuista lisäohjaimista. Itse projekti on **.MAK**-tiedosto. Jos jonkin VBX-tiedoston poistaa projektista, vastaava lisäohjain häviää käytöstä ja sen ikoni häviää työkalulaatikosta.

Ohjelman tekemisessä on kolme päävaihetta:

1. käyttöliittymän piirtäminen,
2. ominaisuuksien arvojen asettaminen ja
3. koodin kirjoittaminen ohjainten tapahtumille.

Seuraavissa luvuissa luodaan esimerkkiohjelma sekuntikello (SEK.MAK) vaihe vaiheelta.



1.3.1 Käyttöliittymän piirtäminen

Käyttöliittymä piirretään tyhjälle lomakkeelle. Työkalulaatikosta valitaan halutut ohjaimet esimerkiksi kaksoisklikkaamalla ohjaimen kuvaa, jolloin ohjain ilmestyy vakiokokoisena lomakkeelle. Tämän jälkeen hiirellä voi muuttaa ohjaimen kokoa ja paikkaa lomakkeella. Sekuntikelloon tarvitaan seuraavat ohjaimet:

otsikko, nappi, ajastin, kuvio, viiva

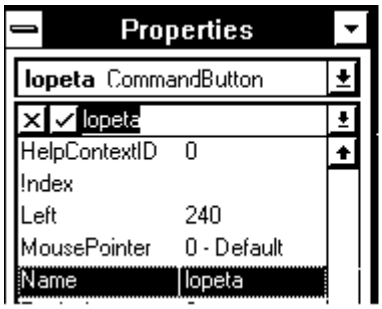
Piirretään lomakkeelle ensin Lopeta-nappi.

1. Kaksoisklikkaa työkalulaatikon nappi-ikonia ja lomakkeelle ilmestyy nappi. Siirrä nappi hiirellä haluamallesi paikalle ja muuta tarvittaessa sen kokoa.

1.3.2 Ominaisuuksien arvojen asettaminen

Kun haluttu ohjain on saatu lomakkeelle, asetetaan sen ominaisuuksille arvot ominaisuusikkunassa. Jokaisella ohjaimen ominaisuudella on tietty oletusarvo, joten aina ei tarvitse muuttaa läheskään kaikkien ominaisuuksien arvoja. Ohjaimen ominaisuuksista tärkein on nimi (Name), jota käytetään koodissa viitattaessa ohjaimeen. Vaikka jokaisella ohjaimella on oletusnimi, se on syytä aina ensimmäiseksi vaihtaa johonkin kuvaavampaan nimeen. Nimissä voi käyttää myös ä:tä ja ö:tä.

2. Aktivoi nappi eli klikkaa sitä hiirellä. Paina pikanäppäintä **F4** tai valitse valikosta **Window - Properties**, niin saat napin ominaisuusikkunan näkyviin.
3. Valitse ominaisuusikkunasta kohta **Name**, kirjoita muokkausriville **lopeta** ja paina enter.



4. Valitse seuraavaksi kohta **Caption** ja kirjoita muokkausriville napin otsikoksi **Lopeta** tai **&Lopeta**, jos haluat liittää nappiin pikanäppäinyhdistelmän Alt - L.

Edellä kuvatulla tavalla asetetaan arvot muidenkin ohjainten ominaisuuksille.

otsikko

Caption: (jätetään tyhjäksi)
Name: **lukema**
BorderStyle: **1 - Fixed Single** (jos halutaan otsikolle kehys)
FontSize: (muuttaa tekstin kokoa)

nappi (x 4kpl)

Caption: **Käyntiin**, **Seis**, **Nollaa**, **Lopeta** (Alleviivaus pikakäyttöä varten saadaan lisäämällä &-merkki kirjaimen eteen.)
Name: **kayntiin**, **seis**, **nollaa**, **lopeta**

ajastin

Interval: **1000** (millisekuntia)
Name: **ajastin**
Enabled: **False** (ettei ajastin käynnisty automaattisesti, kun ohjelma ajetaan)

Ajastin voidaan sijoittaa lomakkeella mihin tahansa kohtaan, koska se ei näy ajon aikana.

kuvio

Name: **kellotaulu**
Shape: **3 - Circle** (ympyrä)

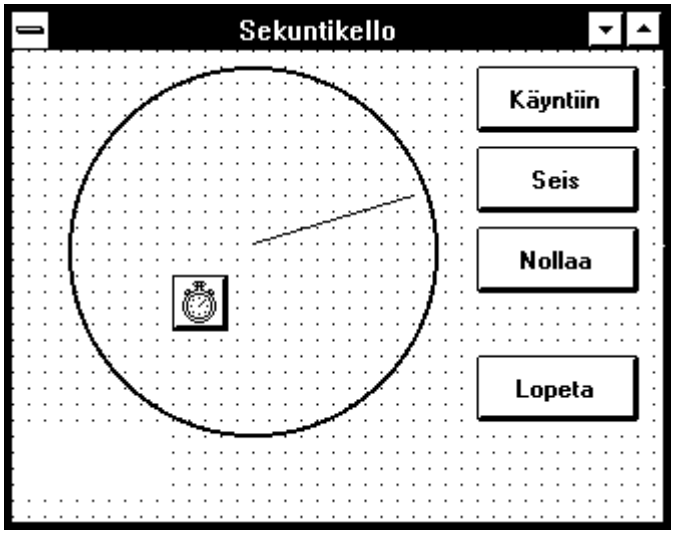
viiva

Name: **sekviisari**
X1, Y1(viivan alkupiste, kellon keskipiste)
X2, Y2(viivan loppupiste)
Varmista, että alku ja loppu ovat juuri näin päin!

lomake

Caption: **Sekuntikello**
Name: **sekuntikello**

MaxButton,	(Jos näille kolmelle ominaisuudelle asetetaan
MinButton,	arvo False, vaikutus näkyy vasta
ControlBox:	ajon aikana.)
BorderStyle:	(Asetus näkyy vasta ajon aikana.)
Picture:	(jos lomakkeelle halutaan taustakuva)

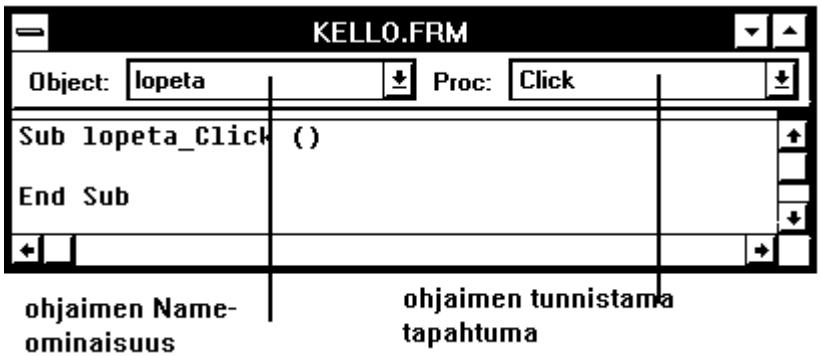


Kun lomakkeita ja projekteja talletetaan, on syytä olla tarkkana. Ensimmäisen kerran projektia talletettaessa VB kysyy nimeä projektiin kuuluville lomakkeille, moduuleille ja itse projektille. Jos lomake on jo kerran talletettu ja se halutaan tallettaa uudella nimellä, on valikosta valittava **Save File As...** Jos valitaan **Save Project As...**, kysytään kyllä projektille uutta nimeä, mutta lomakkeille ja muille projektiin jo kerran talletetuille tiedostoille tarjotaan vain mahdollisuutta tallettaa muutoksen vanhan päälle.

1.3.3 Koodin kirjoittaminen

Kolmas vaihe ohjelman tekemisessä on koodin kirjoittaminen. Aluksi on syytä tarkistaa, että **Options - Environment** -kohdassa on vastattu **Yes** kohtaan **Require Variable Declaration**. Tämä varmistaa sen, että VB vaatii määrittelemään muuttujat. Koodin tallennustavaksi kannattaa valita tekstimuoto. Oletukseksi tämän saa valitsemalla valikosta **Options - Environment** kohdan **Default Save As Format** ja asettamalla siihen **Text**.

Ohjaimeen liittyvä koodi-ikkuna saadaan näkyviin kaksoisklikkaamalla ohjainta. Alla oleva koodi-ikkuna tulee näkyviin kaksoisklikkaamalla lomakkeen Lopeta-nappia.

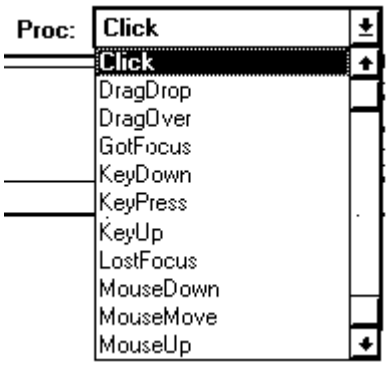


5. Kirjoita aliohjelman aloittavan (Sub) ja lopettavan (End Sub) rivin väliin `End`. Tämä lause lopettaa ohjelman suorituksen, kun painetaan Lopeta-nappia.

```
Sub lopeta_Click()
  End
End Sub
```

Ohjainaliohjelmat muodostuvat automaattisesti ohjaimen nimestä ja tapahtumasta. Jos kuitenkin olet jo kirjoittanut ohjaimelle koodia ja sen jälkeen vaihdat ohjaimen nimeä, löytyy aiemmin kirjoittamasi koodi lomakkeen **general**-osasta.

Jokaisella ohjaimella on oma oletustapahtuma, jonka aliohjelma tulee ensimmäiseksi esiin kaksoisklikkaamalla ohjainta. Napilla oletustapahtuma on Click ja esim. tekstilaatikolla se on Change. Muut ohjaimelle mahdolliset tapahtumat saa esiin koodi-ikkunassa kohdasta **Proc**.



VB:n kommenttimerkki on yksinkertainen lainausmerkki (') ja se merkitsee kommentiksi rivillä kaikki siitä eteenpäin. Jos taas esim. lyhyitä lauseita halutaan kirjoittaa samalle riville, ne erotetaan kaksoispisteellä (:). VB:ssa ei ole mitään merkkiä, jolla voisi katkaista koodirivin ja jatkaa seuraavalta riviltä.

Esimerkkiohjelman sekuntikello siirtää ajastimen avulla viisaria sekunnin välein. Koska ajastimen Enabled-arvo on epätosi, ohjelma odottaa, että käyttäjä painaa Käynnistä-nappia ennen kuin ajastin alkaa toimia. Jos Enabled-arvo olisi tosi, ajastin alkaisi toimia heti, kun ohjelma käynnistetään.

Sekuntikellon lomakekohtaiset muuttujat esitellään lomakkeen **general**-osassa. Tähän osaan pääsee käsiksi kaksoisklikkaamalla lomaketta tai jotakin lomakkeella olevaa ohjainta, jolloin näkyviin tulee koodi-ikkuna. Tämän jälkeen valitaan ikkunan **Object**-kohdasta **(general)** ja samalla **Proc**-kohtaan tulee teksti **(declarations)**. Tähän kohtaan voidaan kirjoittaa halutut lomakekohtaiset muuttujat. Näitä muuttujia voivat käyttää kaikki lomakkeen ohjainaliohjelmat.

6. Kaksoisklikkaa esim. lomaketta ja valitse koodi-ikkunan kohdasta Object kohta (general), jolloin Proc-kohdassa lukee (declarations).
7. Kirjoita tähän kohtaan seuraavat muuttuja- ja vakiomääritykset.

```
Dim sek As Integer
Dim min As Integer              'Jos halutaan lisätä minuuttiviisari.
Dim alku_aika As Long
Dim edellinen_aika As Long
Dim aika As Long
Const PI = 3.1415926535         'vakio
```

Lomakekohtaiset aliohjelmat kirjoitetaan myös general-osaan.

8. Siirry lomakkeen general-osaan. Valitse valikosta **View - New Procedure**. Anna aliohjelmalle tyypiksi **Sub** ja nimeksi **kaanna_viisari**.
9. Kirjoita seuraava koodi valmiiseen pohjaan.

```
Sub kaanna_viisari (v As Line, sek As Integer)
  Dim a As Double
  Dim r As Double
  r = pituus(v)
  a = sek * (2 * PI) / 60 - (PI / 2) 'viisarin kulma radiaaneina
  v.X2 = v.X1 + r * Cos(a)
  v.Y2 = v.Y1 + r * Sin(a)
End Sub
```

10. Valitse jälleen valikosta **View - New Procedure**. Anna nyt aliohjelmalle tyypiksi **Function** ja nimeksi **pituus**. Kirjoita valmiiseen pohjaan alla oleva koodi.

```
Function pituus (l As Line) As Double
  pituus = Sqr((l.X1 - l.X2) ^ 2 + (l.Y1 - l.Y2) ^ 2)
End Function
```

Ohjainkohtaiset aliohjelmat kirjoitetaan suoraan jokaisen ohjaimen omaan koodi-ikkunaan, joka saadaan näkyviin kaksoisklikkaamalla ohjainta. Lomakkeen koodi-ikkuna saadaan näkyviin kaksoisklikkaamalla lomaketta.

```
Sub Form_Load ()
  nollaa_click
End Sub

Sub nollaa_Click ()
  edellinen_aika = 0
  alku_aika = Timer       'Timer-funktio ilmaisee sekunnit, jotka ovat
  aika = 0                'kuluneet keskiyön jälkeen.
  sek = 0
  Call kaanna_viisari(sekviisari, sek)
  lukema.Caption = sek
End Sub

Sub ajastin_Timer ()
  aika = (Timer - alku_aika) + edellinen_aika
  sek = aika Mod 60
  min = aika \ 60         'kokonaisjakolasku
  Call kaanna_viisari(sekviisari, sek)
  lukema.Caption = sek
End Sub

Sub kayntiin_Click ()
  alku_aika = Timer
  ajastin.Enabled = True  'Käynnistetään ajastin.
End Sub

Sub seis_Click ()
  ajastin.Enabled = False        'Pysäytetään ajastin.
  'Senhetkinen aika talteen.
  edellinen_aika = (Timer - alku_aika) + edellinen_aika
End Sub
```

Nyt ohjelma on valmis. **EXE**-versio saadaan, kun valitaan valikosta **File - Make EXE File...**, annetaan ohjelmalle nimi (VB tarjoaa nimeksi projektin nimeä) ja liitetään ohjelmaan haluttu ikoni. EXE-versio vaatii toimiakseen tiedoston **VBRUN300.DLL**, jonka on oltava polussa olevassa hakemistossa. Jos ohjelman tekovaiheessa projektissa on .VBX-

tiedostoja, vaatii valmis EXE-ohjelma toimiakseen, että ne samat .VBX-tiedostot on \WINDOWS\SYSTEM-hakemistossa.

2. Lomakkeet ja muuttujat

2.1 Lomakkeet

Lomakkeet ovat ohjelman käyttäjälle näkyviä ikkunoita. Kun ohjelma käynnistetään, tulee näkyviin aloituslomake ja suoritetaan aliohjelma **Form_Load()**. Aliohjelma **Form_Load()** suoritetaan ennen kuin lomake tulee näkyviin ja vastaavasti ennen kuin lomake poistuu muistista suoritetaan aliohjelma **Form_Unload(***Cancel* **As Integer)**. Metodi **Hide** piilottaa lomakkeen, mutta ei poista sitä muistista. **Show**-metodi lataa lomakkeen muistiin, mikäli se ei jo ole siellä ja näyttää sen sitten. Myös lomakkeen jonkin ominaisuuden arvon muuttaminen koodissa lataa lomakkeen muistiin, mutta ei kuitenkaan näytä sitä.

Ohjelman toinen aloitusvaihtoehto on, että kun ohjelma käynnistetään, aluksi suoritetaan ohjelma **Sub Main**. Tällöin ohjelmoijan on itse huolehdittava, että haluttu lomake tai lomakkeet tulevat näkyviin. **Sub Main** kirjoitetaan projektiin liitettävään moduulitiedostoon; sitä ei voi kirjoittaa lomakkeen general-osaan. Haluttu aloitustapa määrätään valitsemalla valikosta kohta **Options - Project** ja sen jälkeen valitaan kohtaan **Start Up Form** joko haluttu aloituslomake tai vaihtoehto **Sub Main**.

2.2 Muuttujat, datatyypit ja vakiot

VB:n muuttujatyypit ovat **Integer, Long, Single, Double, Currency, String** ja **Variant** (ks. kohta 2.2.1). Esimerkiksi otsikon ja napin Caption-ominaisuus ja tekstilaatikon Text-ominaisuus ovat Variant-tyyppisiä. Muuttujat määritellään sanalla **Dim, Global** tai **Static** (ks. taulukko). Static-muuttujan arvo säilyy aliohjelmakutsujen välillä. Mikäli muuttujalle ei määritellä tyyppiä, se on Variant.

```
Dim x As Integer, y As Integer
```

Varoitus! Määrittely

```
Dim x, y As Integer
```

saattaa aiheuttaa yllätyksen sillä muuttujasta x ei tulekaan kokonaislukumuuttujaa, kuten voisi luulla, vaan Variant-tyyppinen muuttuja!

String on merkkijonomuuttuja, jonka koko muuttuu siihen talletetun merkkijonon koon mukaan. Vakiokokoinen merkkijono määritellään **String *** *koko*.

```
Dim jono As String * 50  'Merkkijonon koko on aina 50 merkkiä.
```

Funktioita merkkijonojen käsittelyyn

Jos funktiosta on kaksi versiota esim. Left ja Left$, niin Left$ palauttaa String-tyypin ja Left Variant-tyypin.

Chr[$](*ANSI-numero*) Numero merkiksi (merkkijonoksi).
Asc(*merkkijono*) Merkin ANSI-arvo.
Val(*merkkijono*) Merkkijono luvuksi (double).
Str[$](*luku*) Luku merkkijonoksi.
StrComp(*jono1, jono2, [tapa]*)
Verrataan kahta merkkijonoa.
Jos *tapa* = 0, "M" < "m".
Jos *tapa* <> 0 "M" = "m".
Jos *tapaa* ei anneta, käytetään oletusvertailua.
Palautusarvot:
-1, jos *jono1* < *jono2*
0, jos *jono1* = *jono2*
1, jos *jono1* > *jono2*
Null, jos *jono1* = Null tai *jono2* = Null
LCase[$](*merkkijono*) Pieniksi kirjaimiksi
UCase[$](*merkkijono*) Isoiksi kirjaimiksi
Len(*mekkijono*) Palauttaa merkkijonon pituuden.

```
Dim a As String
Dim b As String * 4
Dim tulos As Integer
tulos = Len(a)                  'tulos = 0
tulos = Len(b)                  'tulos = 4
a = "kissa"
b = "aa"
tulos = Len(a)                  'tulos = 5
tulos = Len(b)                  'tulos = 4
```

Left[$](*jonolauseke, n*) Palauttaa vasemmalta (alusta) *n* merkkiä

```
Left$("kissa", 2) 'Palauttaa "ki"
```

Right[$](*jonolauseke, n*) Palauttaa oikealta (lopusta) *n* merkkiä.

```
Right$("kissa", 2) 'Palauttaa "sa"
```

Mid[$](*jonolauseke, i, n*) Palauttaa kohdasta *i n* merkkiä.

```
Mid$("kissa", 2, 4) 'Palauttaa "issa"
```

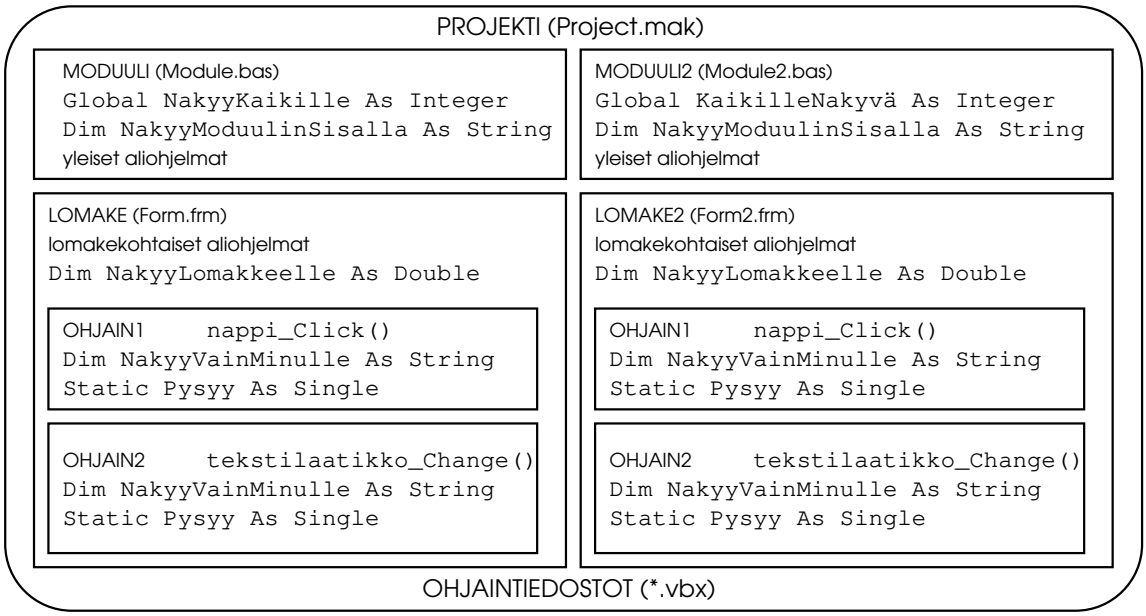
Lomake- ja moduulikohtaiset muuttujat määritellään lomakkeen tai moduulin general -declarations -osassa. **Declarations**-kohdasta löytyy myös teksti **Option Explicit** mikäli valikkokohdassa **Options - Environment** on vastattu **Yes** kohtaan **Require Variable Declaration**. Ohjainaliohjelmien muuttujat määritellään rivien **Sub** ja **End Sub** välissä eivätkä ne eivät näy kyseisen aliohjelman ulkopuolelle.

Määrittely	**Global**	**Static**	**Dim**
ohjainaliohjelma	-	X	x
lomake-declarations	-	-	X
lomakekohtainen aliohj.	-	X	x
moduuli-declarations	X	-	X
moduulialiohjelma	-	X	x

(X = arvo säilyy)

Muuttujat voivat olla tyypiltään myös olio-osoittimia (ks. kohta 7.5) tai käyttäjän määrittelemää tyyppiä (ks. kohta 2.2.2).

Seuraava kaavio havainnollistaa projektin koostumusta ja muuttujien sekä aliohjelmien vaikutusaluetta.



2.2.1 Variant-tyyppi

Variant on muuttujatyyppi, joka voi sisältää mitä tahansa **Integer-, Long-, Single-, Double-, Currency-** tai **String-**tyyppistä tietoa ja lisäksi tyypit **Date, Empty, Null**, mutta ei käyttäjän määrittelemiä omia tietotyyppejä. Tietotyyppien välillä ei tarvita muunnosta vaan Visual Basic hoitaa muunnoksen automaattisesti.

```
Sub nappi_Click ()
  Dim x As Integer
  Dim y As Integer
  x = 8
  y = x + 5
  teksti.Text = x          'Tyypinmuunnosta ei tarvita.
  otsikko.Caption = y
End Sub
```

Variant-muuttujan sisältämän tiedon tyypin voi tarkistaa funktioilla **IsEmpty, IsNull, IsNumeric** ja **IsDate**, jotka palauttavat arvon tosi tai epätosi sen mukaan, onko muuttujassa kysytyn tyyppistä tietoa.

Variant-tyypin käyttäminen merkkijonojen yhteydessä voi aiheuttaa yllättäviä tilanteita yhdistettäessä merkkijonoja +-operaattorilla. Jos kahdessa Variant-muuttujassa on numeerista tietoa, suoritetaan yhteenlasku ja jos kummassakin on merkkijono, suoritetaan yhdistäminen. Mutta jos toisessa muuttujassa on numeerista tietoa ja toisessa merkkijono, syntyy ongelmia. Ensiksi VB yrittää muuntaa merkkijonon numeroksi ja jos se onnistuu, suoritetaan yhteenlasku. Jos muunnos ei onnistu, luodaan uusi merkkijono yhdistämällä numeerinen tieto ja merkkijono. Tämä ongelma voidaan välttää käyttämällä merkkijonojen yhdistämisessä **&**-operaattoria.

```
Dim arvo As Variant        'Variant-tyyppinen muuttuja
arvo = "17"                'Muuttuja sisältää merkkijonon.
arvo = arvo - 15           'Muuttujan arvo on kokonaisluku 2.
arvo = arvo + 1            'Muuttujan arvo on kokonaisluku 3.
arvo = "U" & arvo          'Muuttujan arvo on merkkijono "U3".
arvo = arvo + 1            'VIRHE, koska molemmat muuttujat eivät ole
                           'Variant-tyyppiä.
```

Variant-muuttuja voi sisältää päivämäärä- ja kellonaikatyyppistä tietoa (PAIVA.MAK).

```
Sub nayta_Click ()
  Dim nyky As Variant, aika As Variant, paiva As String
  nyky = Now                                      'Now palauttaa nykypäivän.
                                                  '28.07.1994 7:38:03
  paiva = nyky                                    '"28.07.1994 7:38:03"
  ekapaiva.Text = Format$(paiva, "m/d/yyyy")      '"7.28.1994"
  tokapaiva.Text = nyky                           '"28.07.1994 7:38:03"
  aika = Time                                     'Time palauttaa kellonajan
                                                  '7:39:01
  kello.Text = aika                               '"7:39:01"
End Sub
```

Päivämääriä ja kellonaikoja voi käyttää suoraan koodissa, jos ne ympäröi #-merkeillä.

```
If paivays > #3/6/93# Then
If paivays > #3/6/93 1:20pm# Then
```

Variant-muuttuja on tyhjä (**Empty)**, kunnes siihen sijoitetaan arvo. Jos tällaista muuttujaa käytetään, käsitellään tyhjää arvoa nollana tai tyhjänä merkkijonona riippuen lausekkeesta. Variant-muuttuja voi myös sisältää arvon **Null**. Muihin kuin Variant-tyyppisiin muuttujiin ei voi sijoittaa arvoa **Null**.

2.2.2 Tietueet

Tietueet eli käyttäjän omat tietotyypit määritellään **Type**-lauseella:

```
Type HenkiloTyyppi
  Nimi As String
  Osoite As String
  KenganNumero As Integer
End Type
```

Tämä määrittely on sijoitettava moduulin **declarations**-osaan, koska käyttäjän määrittelemät tyypit ovat aina globaaleja. Määrittelyjä ei siis voi sijoittaa lomaketasolle. Uudentyyppiset muuttujat määritellään kuten muutkin muuttujat (Dim, Static, Global).

```
Dim Herra47 As HenkiloTyyppi, Herra56 As HenkiloTyyppi
```

Tietueen kenttään viitataan samoin kuin olion ominaisuuteen.

```
Herra47.KenganNumero = 56
```

Samantyyppinen tietue voidaan sijoittaa toiseen.

```
Herra47 = Herra56
```

Tietue voi sisältää toisia tietueita ja kiinteitä (ei dynaamisia) taulukoita (ks. kohta 5.2). Tietue ei voi sisältää olio-osoittimia (ks. kohta 7.5).

2.2.3 Vakiot

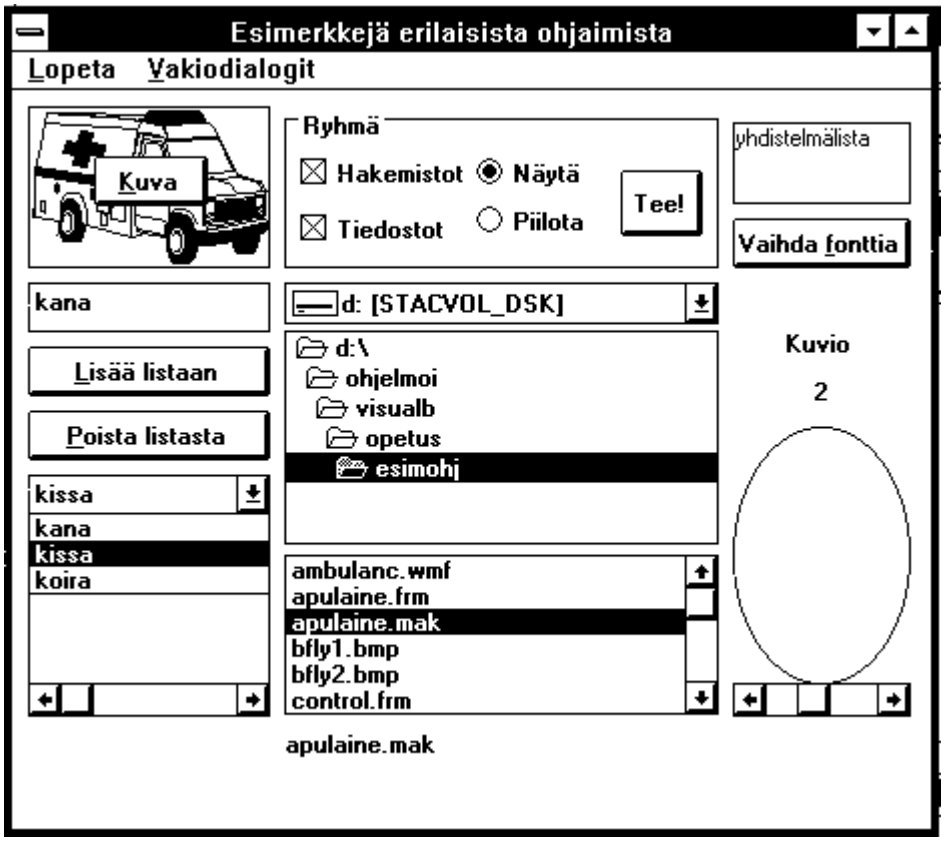
Lomake- ja aliohjelmakohtaiset vakiot määritellään **Const**-lauseella. Jos vakion halutaan näkyvän kaikkialle ohjelmassa, se määritellään moduulissa ja eteen laitetaan sana **Global**.

```
Const PI = 3.14159265
Const VANHAVERSIO = "1.00"
Global Const VERSIO = "3.00"
```

VB:ssa käytössä olevat valmiit vakiot (esim. CF_TEXT, IDOK jne.) on määritelty tiedostossa CONSTANT.TXT ja käyttöön ne saa, kun liittää tämän tekstitiedoston projektiin (**File - Add File...**)

3. Ohjaimista

Esimerkkiohjelma (CONTROL.MAK) esittelee joitakin ohjaimia ja niiden käyttöä. (Projektiin CONTROL.FRM, CMDIALOG.VBX)



3.1 Otsikko (Label)

Otsikko sisältää tekstiä, jota käyttäjä ei voi muuttaa. Otsikkoa voi käyttää mm. selittämään ja nimeämään muita ohjaimia. Otsikolla on kaksi ominaisuutta, jotka vaikuttavat siihen, miten teksti esitetään. Jos

AutoSize **True** Ohjain suurenee vaakasuoraan niin, että haluttu teksti mahtuu kokonaisuudessaan yhdelle riville.

WordWrap **True** Ohjain venyy pystysuunnassa niin, että teksti mahtuu. Leveys muuttuu vain siinä tapauksessa, että jokin yksittäinen sana on pidempi kuin ohjaimen leveys.

Jos halutaan, että WordWrap-ominaisuus on tosi, täytyy myös AutoSize-ominaisuuden olla tosi.

3.2 Tekstilaatikko (TextBox)

Tekstilaatikoita käytetään ottamaan vastaan käyttäjän syöttämää tietoa. Tekstilaatikolla on kaksi keskeistä ominaisuutta: **MultiLine** ja **ScrollBars**. Näitä ominaisuuksia voi muuttaa vain suunnitteluvaiheessa.

MultiLine	ScrollBars	
False	None	Teksti on yhdellä rivillä. Nuolella saa näkyviin ohjaimen leveyden yli menevän tekstin.
False	True	Koska teksti on yhdellä rivillä, vierityspalkkeja ei tule näkyviin.
True	None	Teksti rivittyy automaattisesti ja nuolella saa näkyviin ohjaimen pituuden yli menevän tekstin.
True	Horiz.	Teksti ei rivity automaattisesti vaan se on rivitettävä enterillä.
True	Vert.	Teksti rivittyy automaattisesti.
True	Both	Teksti ei rivity automaattisesti

Vain ajon aikana käytössä olevia ominaisuuksia ovat **SelText, SelStart** ja **SelLength**.

SelText	Merkkijono, joka sisältää tekstilaatikosta valitun (maalatun) tekstin. Jos mitään ei ole valittu, SelText on tyhjä merkkijono.
SelStart	Kokonaisluku, joka ilmaisee, mistä valittu (maalattu) teksti alkaa tai jos mitään ei ole valittu, se ilmaisee kohdan, johon seuraavaksi kirjoitettava teksti tulee.
SelLength	Kokonaisluku, joka ilmaisee valittujen merkkien määrän.

Tekstilaatikon **HideSelection**-ominaisuuden on oltava epätosi, jos halutaan, että valinta näkyy ruudulla.

3.3 Yhdistelmälista (ComboBox)

Yhdistelmälistoja on kolmea tyyppiä:

Dropdown Combo (0)	sallii käyttäjän kirjoittaa listaan muokkauskentän avulla.
Simple Combo (1)	vaatii, että ohjain piirretään riittävän isoksi, jotta kaikki listassa olevat näkyisivät.
Dropdown List (2)	ei salli käyttäjän kirjoittaa suoraan listaan.

Yhdistelmälistaan liittyy ominaisuus **Sorted** eli listassa olevat sanat voidaan järjestää aakkosjärjestykseen. Jos Sorted-ominaisuuden arvo on tosi, lista päivittyy aakkosjärjestykseen automaattisesti, kun siihen lisätään esim. uusi sana tai jos jokin sana poistetaan. Jopa skandinaaviset kirjaimet aakkostuvat oikein.

Esimerkkiohjelmassa (CONTROL.MAK) on kaksi nappia, joiden avulla tekstilaatikkoon kirjoitettu sana lisätään yhdistelmälistaan ja toisaalta taas valittu sana poistetaan yhdistelmälistasta. Yhdistelmälistaan lisätään sanoja metodilla **AddItem**. Jos uuden sanan paikkaa listassa ei anneta, se sijoittuu listan loppuun paitsi, jos listan Sorted-ominaisuus on tosi, jolloin uusi sana menee aakkosjärjestyksessä oikealle paikalleen.

```
Sub lisaa_listaan_Click ()
  lista.AddItem syottorivi.Text
```

```
End Sub
```

Yhdistelmälistasta voidaan poistaa eri tavoin. Jos halutaan poistaa käyttäjän valitsema sana, käytetään **ListIndex**-ominaisuutta. ListIndex on valitun sanan paikan listassa tai -1, jos mitään ei ole valittu. Poisto tapahtuu metodilla **RemoveItem** *index*.

```
Sub poista_listasta_Click ()
  Dim poistettava As Integer
  poistettava = lista.ListIndex
  If poistettava = -1 Then Exit Sub     'Mitään ei ole valittu.
  lista.RemoveItem poistettava
End Sub
```

```
lista.RemoveItem 0 'Poistaa listan ensimmäisen sanan.
```

Jos koko lista halutaan tyhjentää, käytetään **Clear**-metodia. **ListCount**-ominaisuus kertoo sanojen lukumäärän yhdistelmälistassa.

```
Dim maara As Integer
maara = lista.ListCount   'Palauttaa listassa olevien sanojen lukumäärän.
```

3.4 Lista (ListBox)

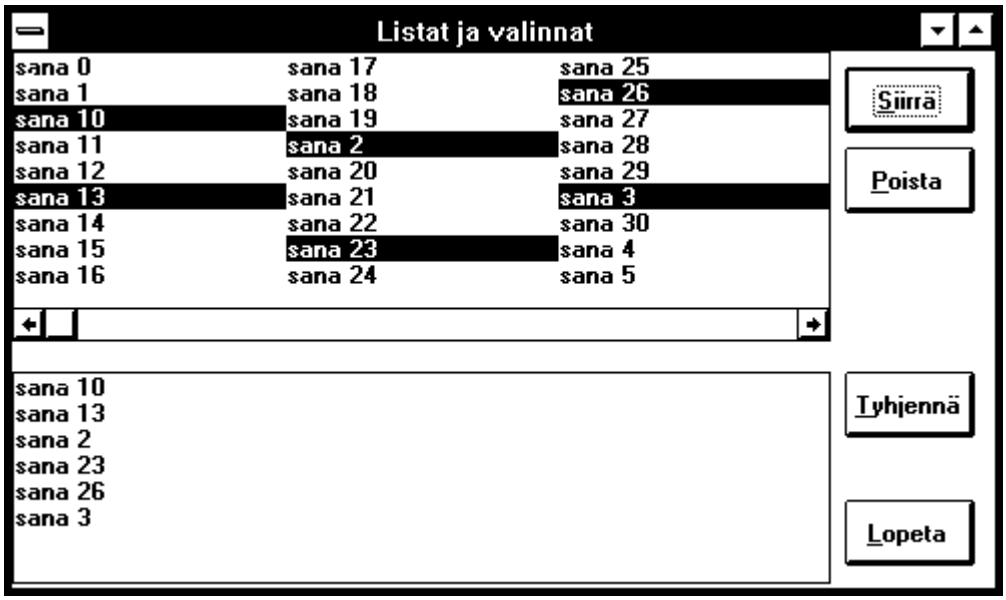
Listaan lisätään ja listasta poistetaan samoilla metodeilla kuin yhdistelmälistastakin. Myös listalla on metodi **Clear** ja ominaisuus **ListCount**. Näiden lisäksi listalla on **Columns**-ominaisuus, jolla määrätään, kuinka monta palstaa listalla on. Ominaisuuden arvoina voivat olla:

0 (yksi palsta, pystysuora vierityspalkki)
1 (yksi palsta, vaakasuora vierityspalkki)
> 1 (haluttu määrä palstoja, vaakasuora vierityspalkki)

Jos halutaan, että käyttäjä voi valita useampia sanoja listasta, asetetaan **MultiSelect**-ominaisuuden arvoksi

1 - Simple (hiirellä klikataan yksitellen)
2 - Extended (hiirellä voidaan vetää valituksi tietty väli).

Alla olevassa esimerkissä (LISTA.MAK) listasta valitaan sanoja klikkaamalla hiirellä yksitellen halutut sanat (eli MultiSelect = 1). Kun painetaan Siirrä-nappia, siirtyvät valitut sanat alempaan listaan.




```
Sub Form_Load () 'ladataan listaan sanoja
  Dim i As Integer
  For i = 0 To 30
    ylalista.AddItem "sana " & i
  Next i
End Sub

Sub siirrä_Click ()
  Dim i As Integer
  For i = 0 To (ylalista.ListCount - 1)       'listassa olevien lkm -1
    If ylalista.Selected(i) = True Then       'jos kohdalla oleva sana   on
                                              'valittu
                                              'List on taulukko listan
                                              'sanoista.
      alalista.AddItem ylalista.List(i)       'lisätään alalistaan

    End If
  Next i
End Sub
```

Poista-Nappi poistaa kaikki ylälistan valinnat kerralla.

```
Sub poistavalinnat_Click ()
  Dim i As Integer
  For i = 0 To (ylalista.ListCount - 1)       'listassa olevien lkm -1
    ylalista.Selected(i) = False              'Poistaa kaikki valinnat.
  Next i
End Sub
```

3.5 Tiedostolistat (DriveListBox, DirectoryListBox, FileListBox)

VB tarjoaa kolme ohjainta tiedostojen ja hakemistojen näyttämiseen: **Levyasema-, hakemisto- ja tiedostolistat**.

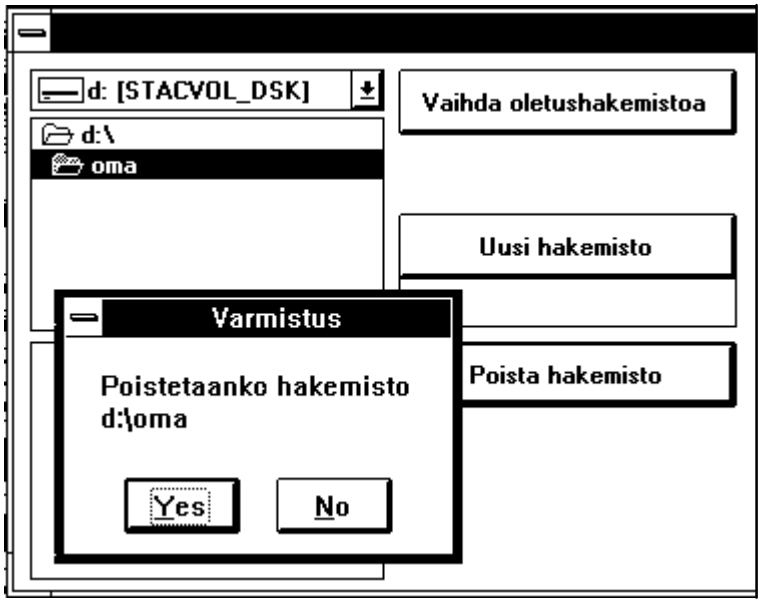
Tarvitaan vain muutama rivi koodia, että nämä kolme ohjainta toimivat yhdessä.

```
Sub levyasema_Change ()
  hakemisto.Path = levyasema.Drive
End Sub

Sub hakemisto_Change ()
  tiedosto.Path = hakemisto.Path
End Sub
```

Tiedostolistaan liittyy ominaisuus **Pattern**, jolla voidaan määrätä, minkä tyyppiset tiedostot näkyvät listassa.

```
tiedosto.Pattern = "*.BMP; *.WMF"
```

Alla oleva esimerkkiohjelma on yksinkertainen tiedostoapulainen (APULAINE.MAK).



Ohjelman avulla voidaan vaihtaa ohjelman oletushakemistoa ja luoda sekä poistaa hakemistoja. Oletushakemisto muutetaan käskyllä **ChDir** ja levyasema käskyllä **ChDrive**.

```
Sub oletus_Click ()
  'Vaihtaa ohjelman oletushakemiston.
  ChDrive levyasema.Drive
```

```
  ChDir hakemisto.Path
  'CurDir$ palauttaa nykyhakemiston.
  nykypolku.Caption = CurDir$
End Sub
```

Uusi hakemisto luodaan käskyllä **MkDir**.

```
Sub uusi_Click ()
  Dim haknimi As String
  If uusihakemisto.Text = "" Then             'Jos hakemistolle ei annettu
    virheenkasittely (-2)                     'nimeä, kutsutaan virheen-
    Exit Sub                                  'käsittelyohjelmaa.
  End If
  If Right$(tiedosto.Path, 1) <> "\" Then     'jos polussa ei \-viivaa
    haknimi = hakemisto.Path & "\" & uusihakemisto.Text
  Else                                        'Talletetaan uuden hakemiston
                                              'nimi polkuineen.
    haknimi = hakemisto.Path & uusihakemisto.Text
  End If
  On Error Resume Next                        'virheenkäsittely ks. kohta
                                              '7.11
  MkDir haknimi                               'Luodaan uusi hakemisto.
  virhe = Err
  On Error GoTo 0
  virheenkasittely virhe
  hakemisto.Refresh                           'Päivitetään hakemistolistaa.
End Sub
```

Hakemisto poistetaan käskyllä **RmDir**.

```
Sub poista_Click ()
  Dim vastaus As Integer, tulos As String
  'Chr (13) ja Chr(10) tekevät rivinvaihdon viestidialogiin.
  tulos = "Poistetaanko hakemisto? " & Chr(13) & Chr(10)
  tulos = tulos & hakemisto.Path
  vastaus = MsgBox(tulos, MB_YESNO, "Varmistus")
  If vastaus = IDYES Then
    On Error Resume Next
    RmDir hakemisto.Path
    virhe = Err       'Otetaan virhenumero talteen.
    On Error GoTo 0   'Laitetaan virheenkäsittely pois päältä.
    virheenkasittely virhe
    hakemisto.Refresh
  End If
End Sub
```

Tiedostojen avaamista ja tallettamista varten on käytössä Windowsin vakiodialogit (ks. kohta 7.8).

3.6 Kuvakehys ja Kuva (PictureBox, Image)

Grafiikkaa voidaan esittää suoraan lomakkeella sekä kuvassa ja kuvakehyksessä.

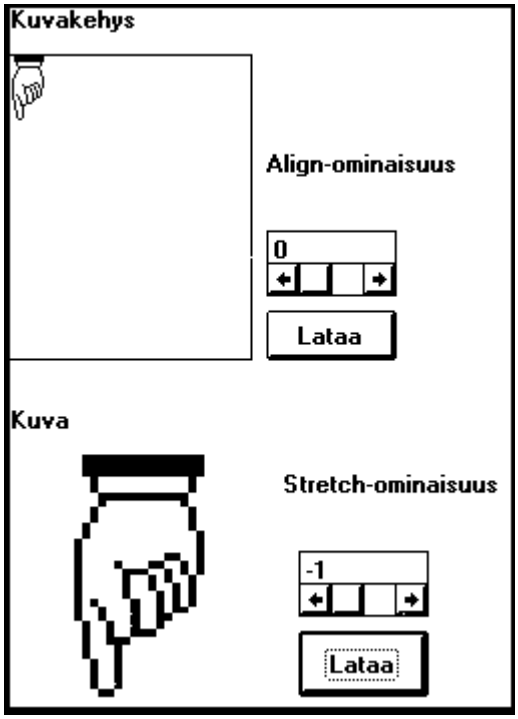
Kuvalla (Image) on **Stretch-** eli venytys-ominaisuus, joka määrää, mukautuuko ohjain kuvan kokoon vai kuva ohjaimen kokoon.

True Kuva mukautuu ohjaimen kokoon
False Ohjain mukautuu kuvan kokoon.

Kuvakehyksellä (PictureBox) on **Align-** eli keskitys-ominaisuus, joka säätelee kuvakehyksen paikkaa lomakkeella. Ominaisuuden arvot ovat:

0	WMF-kuva mukautuu automaattisesti kuvakehyksen kokoon, BMP-kuva ei mukaudu.
1	Kuvavakehys seuraa automaattisesti lomakkeen leveyttä ja sijoittuu lomakkeen yläreunaan.
2	Kuvavakehys seuraa automaattisesti lomakkeen leveyttä ja sijoittuu lomakkeen alareunaan.

Seuraava esimerkki (KUVAT.MAK) havainnollistaa asiaa. Kuvakehykseen ja kuvaan on ladattu ikoni.



Esimerkissä on mahdollista muuttaa kuvakehyksen Align-ominaisuutta ja kuvan Stretch-ominaisuutta ja havaita näiden ominaisuuksien vaikutus. Kuvat ladataan ohjaimiin **LoadPicture**-funktiolla. Kuva ladataan kumpaankin ohjaimeen samalla tavalla.

```
kuvakehys.Picture = LoadPicture("ambulanc.wmf")'Nimeen voi liittyä polku.
```

Kuva poistetaan ohjaimesta samalla funktiolla.

```
kuvakehys.Picture = LoadPicture()
```

Kuvakehykseen (mutta ei kuvaan) voi piirtää graafisilla metodeilla esim. **Line, Circle** tai kirjoittaa metodilla **Print** (ks. liite 5). Kuvakehyksen tai lomakkeen koordinaatistoa voi muuttaa **Scale**-metodin avulla.

3.7 Valintaruudut ja -napit (CheckBox, OptionButton)

Valintaruutuja käytetään ilmaisemaan, onko jokin tietty ominaisuus päällä vai ei. Ne eivät ole riippuvaisia toisistaan, joten yhtäaikaa voi olla rastittuna useita ruutuja. Valintaruudun **Value**-ominaisuus on

0 jos ruutua ei ole valittu,
1 jos ruutu on valittu ja
2 jos kohtaa ei voi valita eli se on harmaa.

Toisin kuin valintaruudut, valintanapit toimivat ryhmänä; vain yksi ryhmästä voi olla valittuna kerralla. Kun valitaan uusi, edellinen valinta poistuu automaattisesti. Valintanapin **Value**-ominaisuus on

True jos nappi on valittu.
False jos nappia ei ole valittu.

Usein valintanappiryhmä sijoitetaan kehykseen. Tällöin on piirrettävä ensin kehys ja vasta sen jälkeen valintanapit kehyksen sisään. Jos halutaan "kehystää" aiemmin piirretyt valintanapit, piirretään kehys, leikataan (Cut) napit ja liitetään (Paste) ne leikepöydältä uuteen kehykseen. Sen ovatko valintanapit tai -ruudut kehyksessä, voi testata vetämällä niitä hiirellä kehyksen reunojen yli. Jos ne saa vedettyä kehyksen ulkopuolelle, ne eivät ole kehyksessä.

Esimerkkiohjelman (CONTROL.MAK) kehyksessä olevat valintaruudut on tehty ohjaintaulukoksi. Kun käyttäjä klikkaa Tee-nappia, tarkistetaan rastit ja toimitaan sen mukaan, onko valittu valintanappi Piilota vai Näytä.

```
Sub tee_Click ()
  If valinta(0).Value = 1 And piilota.Value = True Then 'rastittu
    hakemisto.Visible = False
  Else hakemisto.Visible = True
  End If
  If valinta(1).Value = 1 And piilota.Value = True Then 'rastittu
    tiedosto.Visible = False
  Else tiedosto.Visible = True
  End If
End Sub
```

3.8 Vierityspalkit (ScrollBar)

Esimerkkiohjelmassa (CONTROL.MAK) vaakasuora vierityspalkki on liitetty kuvio-ohjaimeen. Vierityspalkin arvot määräytyvät sille annettujen maksimi- ja minimiarvojen mukaan. Vierityspalkin keskeiset ominaisuudet ovat:

min	Vierityspalkin minimiarvo.
max	Vierityspalkin maksimiarvo
Smallchange	Arvo, joka määrää, miten paljon vierityspalkin arvo muuttuu, kun vierityspalkin nuolta painetaan kerran.
Largechange	Arvo, joka määrää muutoksen suuruuden, kun hiirellä klikataan vierityspalkkia nuolten välillä.
Value	Ilmoittaa vierityspalkin arvon.

```
Sub vierityspalkki_Change ()
  muoto.Shape = vierityspalkki.Value
End Sub
```

3.9 Ajastin (Timer)

Ajastin reagoi kuluvaan aikaan halutuin väliajoin. Jokaisella ajastimella on **Interval**-ominaisuus, joka määrää millisekunteina, kuinka usein Timer-tapahtuma tulee. Interval-ominaisuuden arvo voi olla 0 - 64 767 eli pisin aika kahden Timer-tapahtuman välillä on noin minuutti. Jos jokin ohjelmaa vaatii runsaasti systeemin aikaa, ei Timer-tapahtumia ehkä tule niin usein kuin Interval-ominaisuuteen on asetettu.

Lomakkeella ajastin voi sijaita missä vain, koska se ei ole ajon aikana näkyvissä. **Enabled**-ominaisuus määrää, koska ajastin aloittaa toimintansa. Jos Enabled on tosi, alkaa ajastin toimia heti, kun ohjelma käynnistetään. Asettamalla Enabled epätodeksi, voidaan määrätä, koska ajastin alkaa toimia (esim. kun painetaan jotakin nappia). Ajastinta voi käyttää esimerkiksi kellon tai animaation tekemiseen.

Esimerkkiohjelma on herätyskello (HERATYS.MAK).

Hälytysaika luetaan valmiin syöttödialogin avulla, jonka jälkeen se muunnetaan ajaksi, jos mahdollista.

```
Sub ajastin_Timer ()
  'Asetetaan uusi aika. Funktio Time palauttaa nykyajan.
  kellonaika.Caption = CStr(Time)
  'Hälytetään, jos hälytys on asetettu.
  If halytys <> "" Then
    If Time >= halytys Then
      halytys = ""
      Beep
      MsgBox "Hälytys " & Time, 0, "Herätyskello"
    End If
  End If
End Sub
```

```
Sub halytysaika_Click ()
  Dim kysyttyaika As String
  kysyttyaika = InputBox("Anna hälytysaika:", "Herätyskello", halytys)
  If kysyttyaika = "" Then
    halytys = ""'peruutetaan hälytys
    MsgBox "No ei sitte hälytetä", 0, "Herätyskello"
    Exit Sub
  End If
  'IsDate palauttaa, voidaanko annettu aika kääntää kellonajaksi.
  If Not IsDate(kysyttyaika) Then
    MsgBox "Tää " & kysyttyaika & " o mikää aika.", 0, "Herätyskello"
  Else
    'Muunnetaan annettu aika kellonajaksi.
    halytys = CVDate(kysyttyaika)
  End If
End Sub
```

```
'lomake - declarations
Dim halytys As Variant 'hälytysaika
```

```
Sub Form_Load ()
  halytys = ""
```

```
End Sub
```

3.10 Ominaisuuksien arvot ja niiden muuttaminen (esimerkkinä nappi)

Kaikilla ohjaimilla on erilaisia ominaisuuksia ja niihin liittyviä asetuksia. Ohjaimella on aina nimi, jolla se tunnistetaan ja jota käytetään liitettäessä siihen koodia. Monilla ohjaimilla on ulkonäköön liittyviä ominaisuuksia kuten esim. taustan väri, ohjaimen koko, otsikko, fontti. Näiden lisäksi monilla ohjaimilla on toimintoja määritteleviä asetuksia. Suunnitteluvaiheessa ominaisuuksien arvoja voi muuttaa ominaisuusikkunan avulla ja ajon aikana ohjelmakoodista käsin voidaan tutkia ja muuttaa ohjainten asetuksia (muuttaa esim. napin tekstiä, piilottaa tai näyttää ohjain jne.).

Esimerkiksi napilla on seuraavat ominaisuudet:

```
BackColor           = &H00C0E0FF&
Cancel              =  -1  'True [Onko nappi lomakkeen peruutusnappi.]
Caption             = "&Nappi" [Teksti napissa]
Default             = -1  'True
[Onko nappi lomakkeen oletusnappi. Vain yksi voi nappi olla oletusnappi.]
DragIcon            = "point03.ico"
[Millainen ikoni näytetään kursorina drag-and-drop tapahtumassa.]
DragMode            = 1  'Automatic
[Kun ohjainta klikataan, alkaa raahaaminen automaattisesti.]
                    = 0  'Manual
[Raahaamisen aloittamiseksi täytyy käyttää Drag-metodia.]
Enabled             = 0  'False [Onko nappi pois käytöstä (harmaa) vai ei.]
FontBold            = 0  'False
FontItalic          = -1 'True
FontName            = "Arial"
FontSize            = 13.5
FontStrikethru      = -1 'True
FontUnderline       = -1 'True
Height              = 495 [Napin korkeus]
HelpContextID       = 3 [Avustustiedoston linkkitunnus kun painetaan F1.]
Index               = 0 [Jos kyseessä ohjaintaulukko.]
Left                = 360 [Napin vasemman yläkulman x-suunta.]
MousePointer        = 4 'Icon [Millaiseksi kursori muuttuu napin päällä.]
Name                = nappula [Nimi, jolla nappiin viitataan koodissa.]
TabIndex            = 0    'Määrää, missä järjestyksessä lomakkeen ohjaimet
                           'käydään sarkainnäppäimellä läpi.
[Monesko sarkainnäppäimellä läpikäytävä kohde lomakkeella.]
TabStop             = 0   'False [Pysähdytäänkö sarkainnäppäimellä nappiin.]
Tag                 = "Tähän voi laittaa mitä vain tietoa merkkijonona."
Top                 = 240 [Napin vasemman yläkulman y-suunta.]
Width               = 1215 [Napin leveys]
Visible             = 0   'False [Onko nappi näkyvissä vai ei.]
```

Jokaisella ohjaimella on yksi ominaisuus, jota voi käyttää ilman ominaisuuden nimeä. Esimerkiksi otsikolla se on Caption-ominaisuus, johon voidaan sijoittaa arvo ilman ominaisuuden nimeä:

```
otsikko = "kissa"         'tarkoittaa samaa kuin
otsikko.Caption = "kissa"
```

Tällainen merkintätapa nopeuttaa hieman ohjelman suoritusta mutta koodin luettavuus kuitenkin kärsii.

4. Tapahtumia

4.1 Hiiritapahtumia

4.1.1 MouseUp ja MouseDown

MouseUp- ja MouseDown-tapahtumat syntyvät, kun käyttäjä painaa hiiren napin alas (MouseDown) ja päästää sen taas ylös (MouseUp).

```
Sub kohde_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As
Single)
  PSet (X, Y)       'Värittää pisteen, jossa hiiren nappi painettiin alas.
End Sub

Sub kohde_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As
Single)
  Line -(X, Y)      'Piirtää viivan edellä merkatusta pisteestä siihen
End Sub             'kohtaan, jossa hiiren nappi päästetään ylös.
```

Button-argumentin arvot:

- **1** jos on painettu hiiren vasenta nappia,
- **2** jos on painettu oikeaa nappia ja
- **4** jos on painettu hiiren keskimmäistä nappia.

Shift-argumentin arvot:

- **1** jos Shift-nappi oli alhaalla,
- **2** jos Ctrl oli alhaalla ja
- **4** jos Alt oli alhaalla.

X- ja **Y**-argumentit ilmaisevat hiiren paikan.

4.1.2 MouseMove

MouseMove-tapahtuma saadaan aina, kun hiirtä liikutetaan. Aliohjelman argumentit ovat samat kuin MouseDown- ja MouseUp-tapahtumilla.

```
Sub kohde_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
Single)
  Line -(X, Y)      'Piirtää viivaa hiiren liikkeiden mukaan. Viiva alkaa
                    'vasemmasta yläkulmasta.
End Sub
```

4.1.3 Drag And Drop

Lomakkeella ja ohjaimilla on drag and drop -tapahtuma. Koodissa kirjoitetaan, mitä halutaan tapahtuvaksi, kun jokin ohjain vedetaan määrätyn kohteen päälle. Vedettävien ohjainten **DragMode**-ominaisuus voi olla

automaattinen jolloin ohjainta voi raahata milloin vain tai

manuaalinen jolloin raahaus aloitetaan metodilla **Drag**.

DragIcon on ominaisuus, joka määrää, millaiseksi ohjain muuttuu, kun sitä aletaan raahata. Oletus on ohjaimen kokoinen nelikulmio. Esimerkissä (D_AND_D.MAK) raahattavien ohjainten DragMode on automaattinen. Jos ohjaimen DragMode asetetaan automaattiseksi, ohjaimelta häviää Click-tapahtuma.

Kun DragMode on automaattinen, ohjaimia voidaan raahata koska tahansa, kun ohjelma on käynnissä. Ohjaimen päällä painetaan hiiren nappi alas, ohjain vedetään halutun kohteen päälle ja hiiren nappi vapautetaan. Kohteen DragDrop-tapahtumaan kirjoitetaan haluttu reagoimiskoodi (esim ohjaimen paikkaa siirretään). Esimerkissä ohjaimen paikkaa muutetaan, jos ohjain tiputetaan lomakkeen tyhjään kohtaan.

```
Sub Form_DragDrop (ohjain As Control, X As Single, Y As Single)
'Siirretään vedetty ohjain lomakkeella uuteen paikkaan. Ohjaimen keskikohta
'tulee kursorin kohdalle.
  ohjain.Move (X - ohjain.Width / 2), (Y - ohjain.Height / 2)
End Sub
```

Toisessa esimerkissä (D_AND_D2.MAK) ohjainten DragMode annetaan olla oletuksena eli manuaalisena. Nyt raahaaminen aloitetaan Drag-metodilla, jonka syntaksi on

control.**Drag** *action*

action = 1	kun raahaus aloitetaan (sama kuin käyttäjä painaisi hiiren napin alas)
action = 2	kun raahaus lopetetaan (sama kuin käyttäjä päästäisi hiiren napin ylös)

Metodin avulla raahaus voidaan sallia esim vain tietyissä tapauksissa. Kun ohjaimen DragMode on manuaalinen, siltä häviää Click-tapahtuma vain siksi hetkeksi, kun sitä raahataan. Heti raahauksen loputtua tapahtuma on taas käytössä.

Esimerkissä tekstilaatikon raahaus aloitetaan vasta, kun käyttäjä on painanut A-kirjainta.

```
Sub Form_KeyPress (KeyAscii As Integer)
  'Muutetaan kaikki isoiksi kirjaimiksi.
  Select Case UCase$(Chr$(KeyAscii))
    Case "A"
      teksti.Drag 1
    ...
  End Select
End Sub
```

Esimerkeissä reagoidaan myös tapahtumaan **DragOver**. Tämän tapahtuman avulla voidaan kontrolloida, milloin raahattava ohjain on saapumassa kohteen päälle, on jo kohteen päällä tai on poistumassa kohteen päältä. Esimerkissä ohjaimen saapuminen kohteen päälle aiheuttaa kuvakehyksen taustavärin muuttumisen.

```
Sub kohde_DragOver (ohjain As Control, X As Single, Y As Single, tila As In-
teger)
  Const SAAPUU = 0
  Const LÄHTEE = 1
  Dim vanhatausta As Long
  If tila = SAAPUU Or tila = LÄHTEE Then
    vanhatausta = kohde.BackColor
    kohde.BackColor = kohde.ForeColor
```

```
    kohde.ForeColor = vanhatausta
  End If
End Sub
```

4.2 Näppäintapahtumia

Lomakkeen aktiivinen ohjain ottaa vastaan näppäintapahtuman eli saa viestin, kun käyttäjä painaa jotakin näppäintä. Itse lomake voi ottaa näppäilyjä vastaan vain, jos siinä ei ole yhtään ohjainta tai jos sen kaikki ohjaimet ovat harmaita eli pois käytöstä. Kuitenkin jos lomakkeen **KeyPreview**-ominaisuus on tosi, lomake vastaanottaa kaikki näppäintapahtumat ennen kuin ne menevät sen ohjaimille. Jos tällöin lomakkeen **KeyPress**-ohjelmassa asetetaan parametri **KeyAscii** nollaksi, ei aktiivinen ohjain tunnista näppäintapahtumaa. Tällaisen ohjelman käyttö on hyödyllistä silloin, kun halutaan totettaa aina jokin sama asia, kun tiettyä näppäintä painetaan eikä ole oleellista, mikä ohjain on aktiivinen.

KeyDown	Tapahtuu silloin, kun käyttäjä painaa jotakin nappia.
KeyUp	Tapahtuu silloin, kun käyttäjä vapauttaa painamansa napin.
KeyPress	Tapahtuu VAIN silloin, kun käyttäjä painaa jotakin ASCII-näppäintä (esim. kirjaimet, numerot + BACKSPACE, ENTER, TAB, ESC, SPACE).

4.2.1 KeyUp ja KeyDown

KeyUp- ja KeyDown-tapahtumat

- reagoivat, kun painetaan yhdistelmänäppäimiä (SHIFT, CTRL, ALT).
- reagoivat, kun painetaan nuolinäppäimiä.
- reagoivat, kun painetaan näppäimiä PAGEUP, PAGEDOWN.
- erottavat, onko numero painettu numeronäppäinryhmästä vai kirjainten yläpuolelta - jos NumLock on päällä, niin tulee myös sama KeyPress-tapahtuma kuin kirjainten yläpuoleltakin.
- reagoivat, kun näppäin painetaan alas ja kun se taas vapautetaan - KeyPress reagoi vain näppäimen alaspainamiseen.
- reagoivat, jos painetaan funktionäppäimiä, jotka eivät ole menun pikanäppäimiä (ks. kohta 7.4).

Esimerkkiohjelmalla (NAPPAIN.MAK) voi tutkia näppäintapahtumia.

```
'Tekstilaatikossa painetaan jotakin näppäintä.
'KeyCode on näppäin, jota on painettu ja Shift kertoo, onko painettu myös
'SHIFT, ALT tai CTRL.
'Tulostaa lomakkeelle tapahtuman sekä merkkijonona painetun näppäimen ja
'Shift-arvon.

Sub kohde_KeyDown (KeyCode As Integer, Shift As Integer)
  Dim muutnapit As String
  Select Case Shift
    Case 1
      muutnapit = "SHIFT"
    Case 2
      muutnapit = "CTRL"
    Case 6
      muutnapit = "CTRL+ALT"
    Case 7
      muutnapit = "SHIFT+CTRL+ALT"
```

```
  End Select
  Print "KeyDown " & Chr(KeyCode) & " " & muutnapit
End Sub
```

4.2.2 KeyPress

KeyPress-tapahtuma saadaan, kun käyttäjä painaa jotakin **ASCII**-näppäintä (mm. numerot, kirjaimet). Tapahtuma saadaan myös painamalla

ESC-näppäintä	jos lomakkeella ei ole nappia, jonka **Cancel**-ominaisuus on tosi.
ENTER	jos lomakkeella ei ole nappia, jonka **Default**-ominaisuus on tosi.
BACKSPASE	
TAB	Tapahtuma saadaaan vain, jos kaikki lomakkeen ohjaimet ovat harmaita (pois käytöstä) tai jokaisen ohjaimen **TabStop**-ominaisuus on epätosi.

```
'Tekstilaatikossa painetaan jotakin näppäintä.
'KeyAscii on ASCII-näppäin, jota on painettu.
'Tulostaa lomakkeelle tapahtuman ja merkkijonona painetun näppäimen.

Sub kohde_KeyPress (KeyAscii As Integer)
  Print "KeyPress " & Chr(KeyAscii)
End Sub
```

4.3 Paint-tapahtuma (kuvakehys, lomake)

Windows huolehtii lomakkeiden ja ohjainten uudelleenpiirtämisestä. Jos ominaisuus **AutoRedraw** on epätosi, täytyy ohjelman hoitaa graafisilla metodeilla (**Line, Circle, PSet, Print**) piirrettyjen kuvien päivitys. Paint-tapahtumaa kutsutaan aina, kun osa lomaketta tai kuvakehystä pitää piirtää uudelleen, kun esim. toinen ikkuna on käynyt lomakkeen päällä.

Jokaisella lomakkeella ja kuvakehyksellä (PictureBox) on **AutoRedraw**-ominaisuus. Jos sen arvo on

True	Windows hoitaa myös graafisilla metodeilla piirrettyjen kuvioiden päivityksen.
False	Windows ei hoida graafisilla metodeilla piirrettyjen kuvioiden päivitystä

Jos AutoRedraw on tosi, Paint-tapahtumaa ei tule ja Windows hoitaa päivityksen. Tämä kuitenkin vie muistia, koska jos lomakkeen AutoRedraw on tosi ja lomakkeen kokoakin voi muutella, talletetaan muistiin kuva koko ruudusta. Jos lomakkeesta tehdään vakiokokoinen, talletetaan muistiin vain kuva lomakkeesta. Muistia voi säästää, jos asettaa grafiikan kuvakehykseen ja asettaa kuvakehyksen AutoRedraw-ominaisuuden todeksi ja lomakkeen AutoRedraw-ominaisuuden epätodeksi. Näin muistiin talletetaan vain kuva kuvakehyksen sisällöstä, ei koko lomakkeesta.

Jos lomakkeen tai kuvakehyksen AutoRedraw-ominaisuus on epätosi, pitää viivojen, ympyröiden ym. piirtäminen hoitaa Paint-tapahtumassa. Samoin jos niiden halutaan ilmesty-

vän näkyviin heti, kun lomake tulee näkyväksi, sijoitetaan koodi Paint-tapahtumaan eikä Form_Load-tapahtumaan, koska graafisia metodeja ei toteuteta, mikäli lomake ei ole näkyvä.

5. Ohjelmointikieli

5.1 Kontrollirakenteet

Visual Basicin operaattoreita:

<	pienempi kuin
>	suurempi kuin
=	yhtä suuri kuin
<=	pienempi tai yhtä suuri kuin
>=	suurempi tai yhtä suuri kuin
<>	eri suuri kuin
Mod	jakojäännös
\	kokonaisjakolasku
&	katenaatio
And, Or, Not	

5.1.1 If

```
If x < y Then
  toiminta 1
Elseif x = y
  toiminta 2
Else
  muu toiminta
End If
```

5.1.2 Select Case

```
Select Case jono
  Case "kissa"
    toiminta 1
  Case "koira"
    toiminta 2
  Case Else
    muu toiminta
End Select
```

Case-kohdassa voi olla myös ehtolause. Jos useampi kohta sopii, niin ensimmäinen niistä suoritetaan.

```
Select Case vuosi
  Case < 1000
    toiminta
  Case 1234
    toiminta
  Case 1998, 1711   'vuosi = 1998 tai vuosi = 1711
    toiminta
  Case > 30000
    toiminta
  Case Else
    toiminta
End Select
```

5.1.3 Do...Loop

```
Do While ehto              Do
  toiminta                   toiminta
Loop                       Loop While ehto

Do Until ehto              Do                  =      Do
  toiminta                   toiminta                    toiminta
Loop                       Loop Until ehto             Loop While Not ehto
```

Silmukoista pääsee pois ennen ehdon täyttymistä **Exit Do** -käskyllä.

5.1.4 For...Next

```
For I = 0 To 10 [Step 1]
  toiminta
Next I

For I = 10 To 0 Step -1
  toiminta
Next I
```

Silmukasta pääsee pois ennen ehdon täyttymistä **Exit For** -käskyllä.

5.2 Taulukot

Visual Basicissa taulukot voivat olla yksiulotteisia tai jopa 60-ulotteisia.

5.2.1 Yksiulotteiset taulukot

VB:n taulukot määritellään kuten muuttujatkin (Dim, Staic, Global).

```
Global numerot(14) As Integer
Dim summat(20) As Double
Static sanat(20) As String
```

Ensimmäisessä esimerkkitaulukossa on 15 alkiota (0-14). Oletuksena ensimmäisen alkion indeksi on nolla mutta sen voi muuttaa esim. lisäämällä seuraava rivi lomakkeen / moduulin declaration-osaan:

```
Option Base 1
```

Toinen tapa muuttaa alkuindeksiä on tehdä se taulukon määrittelyn yhteydessä **To**-sanalla:

```
Dim summat(1 To 15) As Double
Dim sanat(100 To 120) As String
```

Summat-taulukon indeksit juoksevat yhdestä viiteentoista ja sanat-taulukon indeksit sadasta sataankahteenkymmeneen. Taulukon alkioihin viitataan: **taulu(***indeksi***)**. Jos koodis-

sa yritetään viitata olemattomaan indeksiin (esim. taulukon alustus tai taulukon käsittely silmukassa), VB antaa virheilmoituksen, kun ohjelmaa yritetään ajaa.

```
Static numerot(1 To 15) As Integer
Dim I As Integer
For I = 1 To 15
  numerot(I) = 0
Next I
```

Taulukon määrittelyssä on mahdollista käyttää myös vakioita. Dynaamisten taulukoiden määrittelyssä voi käyttää muuttujia. Alla on esimerkki venyvästä taulukosta.

```
nappi_Click()
  Static sanat() As String 'dynaaminen taulukko
  Static koko As Integer
  ReDim Preserve sanat(koko) As String        'sanat(0), sanat(1) jne.
                                                '(ks. kohta 7.6)
  sanat(koko) = loota.Text
  koko = koko + 1
End Sub
```

5.2.2 Ohjaintaulukko

Aina kun käyttöliittymää suunniteltaessa samaa ohjainta kopioidaan, Visual Basic tarjoaa mahdollisuutta tehdä ohjaimista taulukko. Ohjaintaulukossa jokaisella ohjaimella tulee olla sama nimi ja ohjainten on oltava samaa tyyppiä (esim. nappeja). Taulukon ohjaimiin viitataan indekseillä, joiden numerointi alkaa nollasta. Jo olemassaolevistakin ohjaimista voi tehdä taulukon vaihtamalla niille kaikille sama nimi ja omat indeksit. Ohjaintaulukkoja ei voi välittää parametreina aliohjelmille.

```
Sub nappi_Click ()  'yksittäinen nappi

End Sub
```

```
Sub nappi_Click (Index As Integer)      'nappitaulukko
                  'Index on painetun napin indeksi.
End Sub           'Vaikka nappeja olisi monta, on niillä vain yksi
                        'ohjainaliohjelma.
```

Ohjaimia voi lisätä taulukkoon ajon aikana **Load**-lauseella. Vain ajon aikana lisättyjä ohjaimia voi poistaa **Unload**-lauseella. Ajon aikana lisätyn ohjaimen **Visible**-arvo on oletuksena epätosi, joten se pitää muistaa muuttaa, jos haluaa ohjaimen näkyvän lomakkeella. Esimerkkiohjelma (LISAAMIN.MAK) lisää lomakkeelle nappeja ja poistaa niitä.

```
Sub lisaa_Click ()
  'Lomakkeella jo oleva nappi määrätään taulukoksi asettamalla
  'sille Index-arvo.
  'Onko lomake matalampi kuin siellä jo olevien nappien korkeus
  '+ lisättävän napin korkeus.
  'Lomakkeella jo olevan napin indeksi on 1.

  If nappi(1).Height * (indeksi + 1) > lisaaminen.ScaleHeight Then
    MsgBox "Tila loppui, venytä lomaketta.", MB_ICONEXCLAMATION, "Nappien
                  lisääminen"
    Exit Sub
  End If
  indeksi = indeksi + 1 'Päivitetään indeksiä.
  Load nappi(indeksi) 'Ladataan uusi nappi.
  'Siirretään uusi nappi vanhan alapuolelle.
```

```
  nappi(indeksi).Move nappi(indeksi - 1).Left, nappi(indeksi - 1).Top +
                              nappi(indeksi - 1).Height
  nappi(indeksi).Visible = True       'Asetetaan uusi nappi näkyväksi.
  nappi(indeksi).Caption = "nappi nro" & indeksi
End Sub
```

```
'lomake - declarations
Dim indeksi As Integer
```

```
Sub poista_Click ()
  If indeksi > 1 Then
    Unload nappi(indeksi) 'Poistetaan nappi.
    indeksi = indeksi - 1 'Päivitetään indeksiä.
  Else
    MsgBox "Lomakkeelle piirrettyä ei voi poistaa.", 48, "Nappien poisto"
  End If
End Sub
```

Ohjaimia ei voi lisätä lomakkeelle ajon aikana muuten kuin taulukkona. Jos halutaan ohjelman jossakin vaiheessa lisätä lomakkeelle vaikkapa uusi nappi, on lomakkeelle suunnitteluvaiheessa piirrettävä yksi nappi ja asetettava sille indeksi (vaikka nolla). Nappi voi olla näkymätön, mutta sen on oltava olemassa, muuten ei uusia nappeja voi lisätä ajon aikana.

5.2.3 Matriisit

Matriisit määritellään seuraavasti:

Esimerkkimatriisi on 10 x 10 matriisi eli indeksit juoksevat nollasta yhdeksään.

```
Dim matriisi(9, 9) As Integer 'matriisi(rivit, sarakkeet)
```

Myös matriisin alkuindeksi voidaan määrätä **To**-lauseella:

```
Dim I As Integer, J As Integer
Static taulukko(1 To 10, 1 To 10) As Double
For I = 1 To 10
  For J = 1 To 10
    taulukko(I, J) = I * 10 + J
  Next J
Next I
```

5.2.4 Dynaamiset taulukot

Dynaaminen taulukko luodaan seuraavasti:

```
[Global | Static | Dim] dyntaulukko() As Double
```

Taulukon koko määritellään ja kokoa muutetaan **ReDim**-lauseella:

```
ReDim dyntaulukko(10, 10)
```

ReDim-lauseella voidaan toistuvasti muuttaa dynaamisen taulukon kokoa mutta ei sen dimensioita. Esimerkiksi jos määritellään dynaaminen taulukko A ja määritellään se kaksiulotteiseksi,

```
Dim A() As Integer
ReDim A(10, 10)
```

ei siitä myöhemmin voi **ReDim**-lauseella tehdä esim. kolmiulotteista. Myöskään taulukon alkioiden tyyppiä ei voi muuttaa ReDim-lauseella. Taulukon kokoa kyllä voidaan muuttaa:

```
ReDim A(20, 20)
```

Avainsanan **Preserve** avulla voidaan taulukon viimeistä dimensiota muuttaa ilman, että taulukosta häviää tietoa. Jos taulukko on yksiulotteinen, voidaan kokoa muuttaa, koska tämä dimensio on viimeinen ja ainoa. Kaksi- tai useampiulotteisissa taulukoissa voidaan muuttaa vain viimeistä dimensiota, ei esimerkiksi ensimmäistä.

```
Dim X() As String
ReDim X(10, 10, 10)
...
ReDim Preserve X(10, 10, 15)
```

Funktioita **UBound** ja **LBound** Syntaksi: **UBound(***taulukonnimi*, *dimensio***)** käytetään tutkittaessa taulukon kokoa. Määritellään esimerkiksi seuraavanlainen taulukko:

```
Dim A(1 To 100, 0 To 3, -3 To 4)
```

UBound-funktion palauttamat arvot taulukosta A:

```
UBound(A, 1) 'palauttaa arvon 100
UBound(A, 2) 'palauttaa arvon 3
UBound(A, 3) 'palauttaa arvon 4
LBound(A, 1) 'palauttaa arvon 1
jne.
```

5.3 Aliohjelmat

Lomakkeen general-osaan kirjoitetut aliohjelmat näkyvät vain lomakkeelle ja niitä voi kutsua vain lomakkeen sisältä, ei esimerkiksi moduulissa olevasta aliohjelmasta. Lomakekohtaiset aliohjelmat ovat siis *private*-tyyppisiä. Koodimoduuleihin kirjoitetut aliohjelmat ovat *public*-tyyppisiä eli ne ovat kaikkien käytössä ja niitä voi kutsua ohjelmassa mistä tahansa. Koska lomakkeen aliohjelmat ovat *private*-tyyppisiä, eri lomakkeilla voi esiintyä samannimisiä aliohjelmia (kaikilla lomakkeilla esim. samanniminen Form_Load -ohjelma). Kahdessa koodimoduulissa ei voi esiintyä samannimisiä aliohjelmia tai muuttujia. Vaikka koodimoduulin aliohjelmat ovat *public*-tyyppisiä, voidaan moduuliin kirjoittaa myös *private*-tyyppisiä aliohjelmia, jotka ovat vain moduulin käytössä.

```
Private Sub ali()
...
End Sub
Private Function summa()
...
End Function
```

Lomakekohtaisia aliohjelmia ei voi mitenkään tehdä *public*-tyyppisiksi paitsi siirtämällä ne moduuliin.

Aliohjelman kaikki muuttujat on mahdollista määritellä **Static**-tyyppisiksi, jos aliohjelman eteen laitetaan sana Static. Tällöin aliohjelman sisäiset muuttujat ovat staattisia vaikka ne olisi määritelty sanalla **Dim**.

```
Static Sub ali()
Static Function aliohjelma() As Integer
```

5.3.1 Sub

Sub-tyyppinen aliohjelma määritellään seuraavasti:

```
Sub aliohjelma (jono As String, luku As Integer)
...
End Sub
```

Sub-tyyppistä aliohjelmaa voi kutsua eri tavoin:

```
aliohjelma "kissa"
Call aliohjelma("kissa")
aliohjelma (otsikko.Caption)     'Sulut, jos parametrina on ominaisuus.
Call ali                         'Ei sulkuja, jos ei parametreja
ali
```

Sub-aliohjelma ei tavallisesti palauta arvoa. Jos aliohjelmassa sijoittaa tuotuun parametriin arvon tai muuten käyttää parametria suoraan aliohjelmassa, se vaikuttaa välittömästi pääohjelman muuttujaan. Tämä voidaan estää **ByVal**-avainsanalla tai laittamalla parametri sulkuihin.

```
ali (jono)   'Nyt aliohjelma ei muuta jonon arvoa.
```

Aliohjelmasta voi poistua ennen sen loppua **Exit Sub** -käskyllä.

Taulukot argumentteina esitellään:

```
Sub lue(taulukko() As String)
Function laske(taulukko() As Integer) As Integer
```

Parametrina taulukko välitetään:

```
lue sanat()
tulos = laske(luvut())
```

5.3.2 Function

Aliohjelman palautustyyppi ilmoitetaan määrittelyn lopussa. Jos funktiolle ei määritellä palautusarvon tyyppiä ja sen nimeen ei sijoiteta palautettavaa arvoa, funktio palauttaa oletuksena **Variant**-tyyppisen tyhjän arvon (**Empty Value**). Jos funktion palautustyypiksi on määritelty merkkijono ja nimeen ei sijoiteta palautusarvoa, funktio palauttaa tyhjän merkkijonon. Jos palautustyyppi on numeerinen, on palautusarvo vastaavassa tilanteessa nolla. Funktion palautustyyppi ei voi olla olio (esim. TextBox) eikä taulukko.

```
Function summa (luku1 As Integer, luku2 As Integer) As Integer
```

```
  summa = luku1 + luku2
End Function
```

Sijoituksia voi aliohjelmassa olla useita, mutta vain yksi jää voimaan. Aliohjelmasta voidaan poistua ennen sen loppua **Exit Function** -käskyllä.

```
Function laske (x As Integer, y As Integer) As Integer
  If x < 0 Then
    laske = x - 10
  ElseIf x = 0 Then  'Palautusarvo on nolla.
    Exit Function
  Else laske = x + y
  End If
End Function
```

Function-aliohjelmaa kutsutaan sijoittamalla tulos muuttujaan.

```
tulos = summa(4, 5)
tulos = funktio()         'Sulut, jos ei parametreja.
```

5.3.3 ByVal

Aliohjelmien parametrinvälitys on oletuksena viittausten (osoitteiden) välittämistä ja tästä syystä on aliohjelmissa mahdollista operoida suoraan pääohjelman muuttujilla. Jos tätä ei haluta sallia, täytyy parametrit välittää arvoina eikä viittauksina. Tämä tapahtuu **ByVal**-avainsanan avulla. Tällöin parametri on kopio alkuperäisestä ja muutokset aliohjelmassa koskevat vain tätä kopiota eivät pääohjelman alkuperäistä muuttujaa.

```
Function aliohjelma (param1 As String, ByVal param2 As Integer) As String
Sub ali (jono As String, ByVal luku As Integer)
```

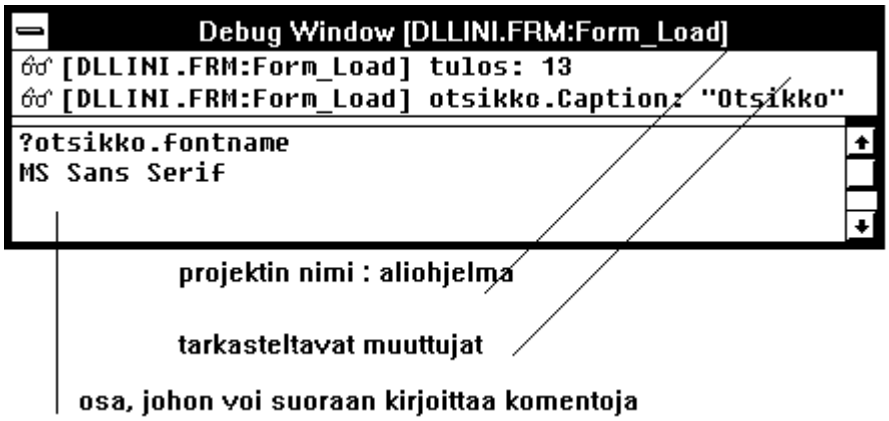
Aliohjelman kutsussa parametrin laittaminen sulkuihin välittää sen arvona. Arvona ei kuitenkaan voi välittää tietuetta, olioa tai taulukkoa.

6. Debuggaus

VB tarkistaa kirjoitetun koodin syntaksin rivi riviltä, jos kohdassa **Options - Environment...** on vastattu **Yes** kohtaan **Syntax Checking** ja ilmoittaa virheistä heti. Ohjelma ajetaan (Run) esimerkiksi painamalla pikanäppäintä F5. Askel askeleelta ohjelma ajetaan esim. painamalla F8.

Tarkasteltavia muuttujia voidaan asettaa kohdasta **Debug - Add Watch...**, jolloin muuttujien arvot näkyvät debug-ikkunassa. Keskeytyskohta ohjelmaan asetetaan siirtämällä kursori halutulle riville ja valitsemalla kohdasta **Debug - Toggle Breakpoint**.

Debug-ikkuna on kaksiosainen.



Ikkunan komento-osa on hyödyllinen esimerkiksi silloin, kun halutaan tutkia lausekkeen arvoa eri sijoituksilla tai jos halutaan tutkia muutenkin erilaisten sijoitusten vaikutuksia tulokseen. Ikkunaan kirjoittaminen aloitetaan ?-merkillä.

Alla olevassa esimerkissä tutkitaan debug-ikkunassa funktion ali palauttamia arvoja eri parametreilla. (Funktio palauttaa merkkijonon, jonka se sitten sijoittaa napin otsikoksi.)

```
?ali(1)
kissa
?ali(9)
koira
?ali(0)
kana
?command1.caption
koira
```

7. Kehittyneemmät ohjelmat

7.1 Useamman lomakkeen ohjelmat

Ohjelma voi koostua useista lomakkeista ja dialogeista (dialogitkin ovat lomakkeita). Dialogeja voi tehdä itse tai sitten voi käyttää valmiita dialogeja (ks. kohta 7.8). Uusi lomake lisätään projektiin valitsemalla valikosta **File** - **New Form**.

Ohjelmassa lomake näytetään metodilla **Show**.

[*Form.***]Show [***style***]**

style = 0	lomake on **modeless**.
style = 1	lomake on **modal**.

Jos tyylin arvoa ei anneta, se on oletuksena nolla.

Jos lomake on **modal**-tyyppinen, käyttäjä ei voi aktivoida ohjelman muita lomakkeita ennen kuin on sulkenut modal-lomakkeen. **Modeless**-lomake sallii muidenkin lomakkeiden aktivoimisen.

```
lomake.Show 1       'Lataa ja näyttää modal-lomakkeen.
```

Load	lataa lomakkeen muistiin.
Unload	poistaa lomakkeen muistista.
Show	näyttää lomakkeen ja lataa sen tarvittaessa muistiin ennen näyttämistä.
Hide	piilottaa lomakkeen, mutta ei poista sitä muistista.

Lomakkeen nappeihin liittyvät ominaisuudet **Default** ja **Cancel**.

Default

- Vain lomakkeen yhden napin Default-ominaisuus voi olla tosi.
- Tämän napin Click-tapahtuma suoritetaan, kun käyttäjä painaa **enter**iä.
- Tämän napin **TabIndex** täytyy olla **nolla**, jos halutaan, että lomakkeen avautuessa nappi on aktiivinen.

Cancel

- Vain lomakkeen yhden napin Default-ominaisuus voi olla tosi.
- Tämän napin Click-tapahtuma suoritetaan, kun käyttäjä painaa **esc**-näppäintä.

Jos lomakkeen avautuessa halutaan asettaa jokin tietty ohjain aktiiviseksi, asetetaan tämän ohjaimen **TabIndex**-ominaisuuden arvoksi nolla. Aktiivisuus voidaan asettaa myös metodilla **SetFocus** syntaksi: **(***ohjain***.SetFocus)**.

Toisella lomakkeella oleviin ohjaimiin ja niiden ominaisuuksiin viitataan huutomerkin (!) avulla.

```
toinenlomake!otsikko.Caption = nimi.Text
```

7.2 Globaalit Forms- ja Controls-taulukot

VB pitää kirjaa ohjelman lomakkeista ja lomakkeiden ohjaimista. Jokaisessa ohjelmassa on yksi globaali lomaketaulukko **Forms** ja jokaisella lomakkeella on oma ohjaintaulukkonsa **Controls**.

7.2.1 Forms-taulukko

Koska Forms-taulukko on globaali, siihen voi viitata mistä tahansa koodin kohdasta. Sitä mukaa kun ohjelmassa ladataan lomakkeita muistiin, ne ilmestyvät myös Forms-taulukkoon. Kun lomake poistetaan muistista, se poistuu myös Forms-taulukosta. Lomakkeilla ei ole taulukossa mitään tiettyä paikkaa, koska taulukko päivittyy (tiivistyy) aina, kun jokin lomake poistuu muistista. Forms-taulukolla on **Count**-ominaisuus, joka ilmoittaa lomakkeiden lukumäärän taulukossa. Forms-taulukkoon viitataan samoin kuin muihinkin taulukoihin **Forms(***index***)**. **Forms.Count** palauttaa lomakkeiden määrän taulukossa ja sitä voi käyttää esim. silmukassa, kun kaikki ladatut lomakkeet täytyy käydä läpi.

```
For i = 0 To Forms.Count -1
  Forms(i) ...
Next i
```

7.2.2 Controls-taulukko

Jokaisella lomakkeella on oma Controls-taulukko. Taulukko sisältää kaikki lomakkeen ohjaimet. Jos lomakkeella on ohjaintaulukkoja, jokainen ohajaintaulukon alkio lisätään Controls-taulukkoon. Ohjainten järjestys taulukossa ei ole pysyvä vaan muuttuu sitä mukaa, kun ohjaimia ladataan tai poistetaan (Load, Unload). Taulukkoon viitataan indeksillä **[***form***!]Controls(***index***)**. Myös Controls-taulukolla on **Count**-ominaisuus, jota myös voi käyttää esim. silmukoissa.

```
For i = 0 To lomake.Controls.Count - 1
  lomake!Controls(i) ...
Next i
```

7.3 Lomakkeen esiintymät

Lomakkeen uusi esiintymä luodaan sanalla **New**, jota on mahdollista käyttää vain lomakkeen yhteydessä.

```
Dim uusi As New aloituslomake  'Aloituslomake täytyy olla jo olemassa.
```

Jos luodaan ohjelman aloitusikkunasta uusi esiintymä, saadaan samalla koko ohjelmasta uusi esiintymä.

```
Sub nappi_Click()
  Dim F As New aloituslomake
  F.Show  'Ladataan ja näytetään uusi esiintymä.
End Sub
```

Esimerkin F on paikallinen muuttuja, mutta aliohjelman suorituksen jälkeen tämä lomakkeen uusi esiintymä säilyy kunnes se suljetaan tai poistetaan muistista (Unload). Uuden lomakemuuttujan F (ja muidenkin olio-osoittimien (ks. kohta 7.5)) arvo on **Nothing** kunnes siihen kohdistetaan jokin toiminta tai sen jonkun ominaisuuden arvoa muutetaan. Globaaleihin, lomaketason tai paikallisiin staattisiin olio-osoittimiin on hyvä sijoittaa Nothing, kun niitä ei enää käytetä, sillä tämä vapauttaa muistia. Paikallisiin olio-osoittimiin VB sijoittaa automaattisesti Nothing, kun aliohjelman suoritus loppuu.

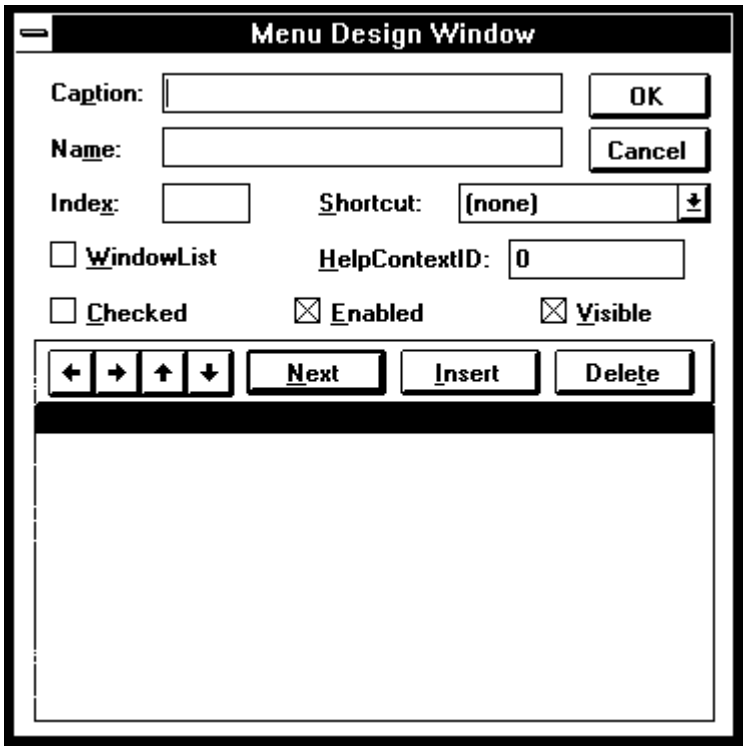
- Lomakkeen jokainen uusi esiintymä on kopio suunnitteluvaiheessa luodusta lomakkeesta.
- Jokaisen esiintymän ominaisuuksien arvoja voidaan muuttaa sen vaikuttamatta muihin esiintymiin.
- Lomakkeen kaikki esiintymät jakavat saman koodin mutta jokaisella esiintymällä on oma kopionsa muuttujista.
- Vain globaaleja muuttujia voidaan käyttää yhteisinä kaikille esiintymille.
- Avainsanalla **Me** voidaan viitata lomakkeen senhetkiseen esiintymään, joka ei aina välttämättä ole aktiivinen lomake.

```
Sub sulje_Click()
  Unload Me  'Poistaa muistista senhetkisen esiintymän.
End Sub
```

Me on mahdollista välittää parametrina aliohjelmalle, joka hyväksyy argumentikseen lomakkeen.

7.4 Valikot (Menu)

Valikot luodaan **Menu Design** -ikkunassa (**Window - Menu Design**).

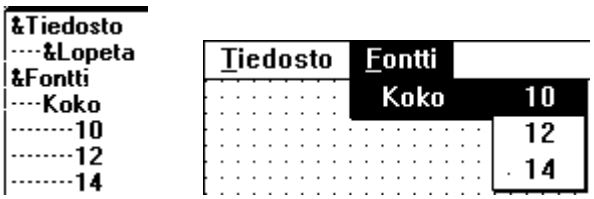


Caption	Kohtaan kirjoitetaan se teksti, jonka halutaan näkyvän käyttäjälle valikossa.
Name	Valikkokohdan koodinimi.
Index	Kohdan tunnus, jos kyseessä on taulukko.
Shortcut	Määrittää valikkokohdalle pikanäppäinyhdistelmän, jos sellainen halutaan antaa.
WindowList	Näytetäänkö valikossa lapsi-ikkunat (ks. kohta 7.13)
Checked	= True, eli kohdassa rasti, valikkokohdan eteen tulee väkänen.
Enabled	= False, valikkokohta on harmaa.
Visible	= False, valikkokohta on piilossa.

Ominaisuuksien **Caption, Checked, Enabled** ja **Visible** arvoja voi muuttaa ajon aikana. Nuolien avulla määrätään valikkokohtien suhteet toisiinsa. Napeilla **Next, Insert, Delete** lisätään ja poistetaan valikkokohtia.

Luodaan malliksi valikko:

1. Klikkaa haluamasi lomake aktiiviseksi. Valitse valikosta **Window - Menu Design**.
2. Kirjoita **Caption**-riville *&Tiedosto* ja anna nimeksi vaikka *menu_tiedosto*.
3. Lisää seuraavaksi Tiedosto-valikolle alakohta. Paina **Next**-näppäintä, kirjoita **Caption**-kohtaan *&Lopeta*, nimeksi *tiedosto_lopeta* ja pikanäppäinyhdistelmäksi *Ctrl+L*.
4. Paina seuraavaksi nuolinappia . Pisteviiva ilmaisee, että Lopeta on Tiedoston alakohta.
5. Tehdään seuraavaksi alavalikko fontin koolle. Tee ensin edellä kuvatulla tavalla Fontti-valikko ja sille alakohta Koko. Alavalikko Koko-kohdalle saadaan painamalla sitä seuraavan valikkokohdan kohdalla nuolta oikealle kaksi kertaa.



Menukohdat voidaan kirjoittaa taulukoksi antamalla niille sama nimi ja indeksit. Yllä olevassa esimerkissä kohdat 10, 12, 14 ovat taulukkona nimeltä koko ja indeksit ovat 0, 1, 2.

Valikkokohtien erotin tehdään kirjoittamalla Caption-kohtaa yksi tavuviiva (-). Myös erottimelle on annettava nimi.

Valikoilla ei ole kuin yksi tapahtuma: **Click**. Koska fonttien kokovaihtoehdot ovat taulukkona, **Index** kertoo, mitä kohtaa menussa on klikattu.

```
Sub koko_Click (Index As Integer)
  teksti.FontSize = Val(koko(Index).Caption)
End Sub
```

7.5 Olio-osoittimet

Visual Basic -ohjelmassa on mahdollista määritellä myös olio-osoittimia. Kuten muillakin muuttujilla myös olio-osoittimilla on tyyppi. Kuitenkin toisin kuin muilla muuttujilla, olio-osoittimilla on kaksi erikoistyyppiä: **yleinen (Generic)** ja **erityinen (Specific)**. Yleisiä olio-osoitintyyppejä on kolme:

Form	Muuttujaan voidaan sijoittaa mikä tahansa ohjelman lomake.
Control	Muuttujaan voidaan sijoittaa mikä tahansa ohjelman ohjain.
MDIForm	Ohjelman MDI-lomake, jos sellainen on. (ks. kohta 7.13)

```
Dim lomake As Form        'osoitin mihin tahansa lomakkeeseen
Dim ohjain As Control     'osoitin mihin tahansa ohjaimeen
```

Muun muassa seuraavat ovat mahdollisia erityisiä olio-osoitintyyppejä:

CheckBox	HScrollBar	PictureBox
ComboBox	Image	Shape
CommandButton	Label	TextBox
DirListBox	Line	Timer
DriveListBox	ListBox	VScrollBar
FileListBox	Menu	
Frame	OptionButton	

```
Dim tekstiloota As TextBox       'osoitin tekstilaatikkoon
```

Yleiseen ja erityiseen olio-osoittimeen sijoitetaan arvoja **Set**-lauseen avulla.

```
Dim uusiohjain As Control 'osoitin mihin tahansa ohjaimeen
Dim uuskuva As PictureBox 'osoitin kuvakehykseen
Set uusiohjain = otsikko  'uusiohjain osoittaa jo olemassaolevaan ohjaimeen.
Set uuskuva = kuva        'uuskuva osoittaa jo olemassaolevaan ohjaimeen.
```

Olio-osoitin on siis osoitin, joka Set-lauseella laitetaan osoittamaan haluttuun olioon.

```
Dim x As Control                        'olio-osoitin
Set x = alkulomake!tekstilaatikko       'x osoittaa alkulomakkeen
                                              'tekstilaatikko-olioon.
x.Text = "kissa"                        'Tämän sijoituksen jälkeen

      'alkulomake!tekstilaatikko.Text =
            '"kissa".
```

Jos osoitin x on tyyppiä Control ja osoitin y tyyppiä TextBox, voidaan tehdä sijoitus `Set x = y`, mutta ei päinvastoin.

Is-operaattorilla voidaan verrata kahta olio-osoitina toisiinsa eli osoittavatko ne samaan kohteeseen.

```
tulos = olio1 Is olio2    'tulos = True (-1), jos oliot osoittavat samaan
                                 'kohteeseen, muuten False (0).
If olio1 Is Nothing Then  'Onko osoitin "tyhjä".
```

Oliot välitetään aliohjelmille aina viittauksina. Jos aliohjelman argumenttina on esim. yleinen oliotyyppi Control, voi kutsussa olla parametrina mikä tahansa ohjain (nappi, lista

jne.). Seuraava esimerkki muuttaa minkä tahansa ohjaimen Visible-ominaisuuden päinvastaiseksi (jos ohjaimella on tämä ominaisuus).

```
Sub vaihda_nakyvyys(ohjain As Control)
  ohjain.Visible = Not ohjain.Visible
End Sub
```

Kun aliohjelman argumenttina on yleinen oliotyyppi esim. Control, voi olla tarpeen selvitää mikä ohjain aliohjelmalle kulloinkin välitetään. Tämä tapahtuu erityisellä **If...TypeOf** -lauseella. Alla oleva esimerkki testaa, mitä kuvakehyksen päälle on vedetty.

```
Sub kohdekuva_DragDrop(ohjain As Control, x As Single, y As Single)
  If TypeOf ohjain Is Image Then
    kohdekuva.Picture = ohjain.Picture
  ElseIf TypeOf ohjain Is TextBox Then
    kohdekuva.Print ohjain.Text 'Tulostaa tekstilaatikon tekstin kuvaan.
  End If
End Sub
```

Jokainen ohjaintyyppi on testattava erikseen. Ei ole siis mahdollista kirjoittaa:

```
If TypeOf ohjain Is Image Or TypeOf ohjain Is PictureBox Then
```

Vaan täytyy kirjoittaa:

```
If TypeOf ohjain Is Image Then
  toiminta
End If

If TypeOf ohjain Is PictureBox Then
  toiminta
End If
```

Operaattoria **Not** ei voi käyttää **If...TypeOf** -lauseessa eikä lausetta **TypeOf** voida käyttää **Select Case**:n yhteydessä.

7.6 Olio-osoitintaulukot

Olio-osoitintaulukko luodaan kuten muutkin taulukot käyttämällä **Dim-, Static-, Global-** ja **ReDim**-lauseita. Olio-osoitintaulukot voivat olla siis myös dynaamisia taulukoita.

```
Dim kuvat(10) As Image    'Voi sisältää vain kuvia.
Dim kuvia(10) As Control  'Voi sisältää mitä vain ohjaimia.
```

Olio-osoitintaulukon ja ohjaintaulukon eroja:

Toiminta	**Oliotaulukko**	**Ohjaintaulukko**
Taulukon luonti	Dim, ReDim, Static, Global	Sama nimi ja indeksit.
Muuta kokoa	ReDim, Preserve	Load, Unload
UBound, LBound	kyllä	ei
Vastine ruudulla	jos osoittimeen sijoitetaan	aina

Olio-osoitintaulukkoon sijoitetaan arvoja **Set**-lauseen avulla.

```
Dim oliot(2) As Control
```

```
Set oliot(0) = otsikko  'jo olemassaoleva olio
```

7.7 Taulukko-ohjain (Grid)

Taulukko-ohjain on ruudukko, jonka rivien ja sarakkeiden määrää voi muuttaa suunnitteluaikana ja ajon aikana. Ohjaimen valmiisiin ominaisuuksiin kuuluu koko sarakkeen tai rivin maalaaminen, kun hiirellä klikataan kiinteää riviä tai saraketta. Käyttäjä ei pääse suoraan kirjoittamaan taulukon soluihin vaan käyttäjän antama tieto täytyy välittää esim. syöttödialogin avulla. Taulukon solut voivat sisältää yhtäaikaa kuvia ja tekstiä.

Rows	Ominaisuus ilmaisee taulukon rivien määrän.
Cols	Ominaisuus ilmaisee taulukon sarakkeiden määrän.
FixedRows	Ilmaisee kiinteiden rivien määrän.
FixedCols	Ilmaisee kiinteiden sarakkeiden määrän.

Taulukon määrättyyn soluun kirjoitetaan tekstiä tai liitetään kuva asettamalla ensin osoitin osoittamaan haluttuun kohtaan ja sen jälkeen käytetään taulukon **Text**- ja **Picture**-ominaisuuksia. Sarake nolla on aina ensimmäinen kiinteä sarake ja vastaavasti rivi nolla ensimmäinen kiinteä rivi. Taulukossa pitää olla vähintään yksi kiinteä rivi ja yksi kiinteä sarake.

```
taulukko.Col = 1    'Asetetaan sarakeosoitin.
taulukko.Row = 1    'Asetetaan riviosoitin.
taulukko.Text = "rivi 1, sarake 1"
taulukko.Picture = LoadPicture(".\face02.ico")
```

Taulukkoon lisätään rivejä **AddItem**-metodilla, jonka syntaksi on *ohjain.***AddItem** *teksti, indeksi.* Jos lisättävän rivin indeksi jätetään antamatta, rivi lisätään taulukon loppuun.

```
taulukko.AddItem "Saldo:" 'Lisää taulukon loppuun uuden rivin ja laittaa
                                  'sarakkeeseen nolla tekstin "Saldo:".
```

Rivejä poistetaan metodilla **RemoveItem**, jonka syntaksi on *ohjain***RemoveItem** *indeksi*. Jos poistettava rivi eli indeksi jätetään antamatta, poistetaan taulukon viimeinen rivi.

Sarakkeita ei voi lisätä kuin taulukon loppuun.

```
taulukko.Cols = taulukko.Cols + 1
```

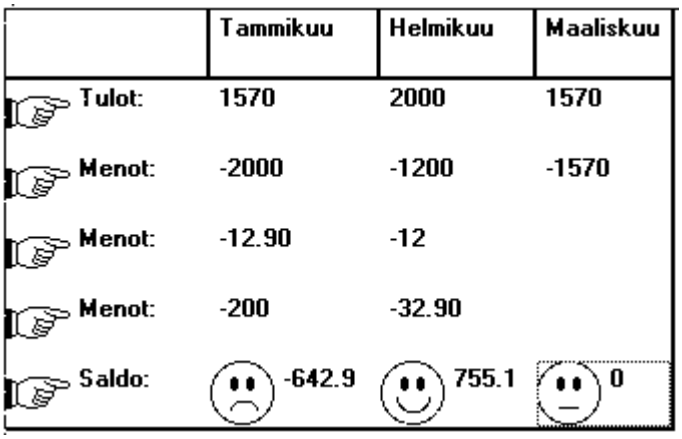
Sarakkeita ei myöskään voi poistaa muuta kuin taulukon lopusta.

```
taulukko.Cols = taulukko.Cols - 1
```

Koodista käsin voidaan maalata halutut solut seuraavien ominaisuuksien avulla:

SelStartCol	Määrää, mistä sarakkeesta maalaus aloitetaan.
SelEndCol	Määrää, mihin sarakkeeseen asti maalataan.
SelStartRow	Määrää, miltä riviltä maalaus aloitetaan.
SelEndRow	Määrää, mille riville asti maalataan.

Esimerkkiohjelma (RUUDUKKO.MAK, projektiin RUUDUKKO.FRM ja GRID.VBX) valvoo menoja ja tuloja kuukausittain. Taulukkoon voi lisätä rivejä ja sarakkeita sekä poistaa niitä. Saldo lasketaan riveiltä 1— taulukko.Rows - 2.



7.8 Vakiodialogit

VB tarjoaa viisi valmista dialogia: tiedoston avaus ja talletus, fontin vaihto, värin vaihto ja tulostus. Dialogit eivät ole saatavilla suunnitteluvaiheessa vaan ainoastaan ajon aikana. Valmiit dialogit saadaan ohjelmaan mukaan, kun projektiin liitetään tiedosto CMDIALOG.VBX ja lomakkeelle piirretään dialogiohjain . Ohjaimen voi sijoittaa lomakkeella mihin tahansa, koska se ei näy ajon aikana. Ohjaimia voi olla samassa ohjelmassa useita. Ohjaimen **Action**-ominaisuus määrää, mikä dialogi kulloinkin näytetään.

vakiodialogi.Action = 1	Tiedoston avaus
vakiodialogi.Action = 2	Talleta nimellä
vakiodialogi.Action = 3	Värin vaihto
vakiodialogi.Action = 4	Fontin vaihto
vakiodialogi.Action = 5	Tulostus

7.8.1 Tiedoston avaus (Open)

Tiedoston avausdialogi saadaan näkyviin, kun Action-ominaisuuden arvoksi asetetaan yksi.

```
valmisdialogi.Action = 1 'Tämä avaa dialogin näytölle.
```

Dialogin ominaisuuksien arvot on asetettava ennen dialogin näyttämistä, jotta ne tulisivat voimaan.

DialogTitle	Määrää dialogin otsikon. Oletuksena avausdialogilla se on Open.
Filter	Ominaisuudella voidaan määrätä, mitkä tiedostot näytetään tiedostolistassa (vrt. tiedostolista.Pattern) ja mikä selitys näkyy tiedostotyyppilistassa.
FilterIndex	Jos suodattimia annetaan useampia, tällä ominaisuudella määrätään se, joka näytetään oletuksena. Alla olevassa esimerkissä oletuksena näytetään .txt-tiedostot.

CancelError Jos ominaisuuden arvo on tosi, saadaan cancel-napin painamisesta virhe numero 32755. Jos ominaisuuden arvo on epätosi, ei cancelin painamisesta saada tietoa.

```
avaa.DialogTitle = "Avaa..."
avaa.Filter = "Teksti (*.txt)|*.txt|Kaikki (*.*)|*.*"
avaa.FilterIndex = 1
                                    'HUOM asetukset on tehtävä ennen kuin dia-
logi
                                    'näytetään!!!
On Error Resume Next                'Jos tulee virhe, jatka seuraavasta kohdas-
ta.
avaa.Action = 1                     'Näytetään avaa-dialogi.
virhe = Err                         'Otetaan virhenumero talteen.
On Error GoTo 0                     'Virheenkäsittely pois päältä.
If virhe > 0 Then Exit Sub          'Jos virhe = 32755, painettiin Cancel.
                                    'kasiteltava_tiedosto-muuttujaan tallete-
taan
                                    'valitun tiedoston nimi ja polku.
kasiteltava_tiedosto = avaa_talleta.Filename
If kasiteltava_tiedosto <> "" Then
  onnistui = avaa_tiedosto(kasiteltava_tiedosto)
...
```

InitDir Ominaisuudella voidaan määrätä, mikä hakemisto dialogissa näytetään oletuksena. Jos hakemistoa ei määrätä, näytetään senhetkinen hakemisto.

FileName Ominaisuudella palautetaan käyttäjän valitsema tiedostonimi polkuineen. Tiedoston nimi palautetaan, kun käyttäjä painaa dialogin OK-nappia.

```
Dim avattava As String
avattava = valmisdialogi.Filename'Palauttaa valitun tiedoston polkuineen.
```

7.8.2 Tiedoston talletus nimellä (Save As)

Tiedoston talletusdialogi saadaan näkyviin, kun **Action**-ominaisuuden arvoksi asetetaan kaksi.

```
valmisdialogi.Action = 2
```

Talletusdialogilla on samat ominaisuudet kuin avausdialogillakin. **FileName**-ominaisuus palauttaa käyttäjän talletettavalle tiedostolle antaman nimen. Jos tiedosto on jo olemassa, kysytään, talletetaanko päälle.

7.8.3 Värin vaihto (Color)

Värinvaihtodialogi saadaan näkyviin, kun **Action**-ominaisuuden arvoksi asetetaan kolme.

```
valmisdialogi.Action = 3
```

Color-ominaisuus sisältää käyttäjän valitseman värin. Esimerkiksi lomakkeen taustaväriä muutetaan lauseella

```
lomake.BackColor = valmisdialogi.Color
```

7.8.4 Fontin vaihto (Font)

Fontinvaihtodialogi saadaan näkyviin, kun **Action**-ominaisuuden arvoksi asetetaan neljä.

```
valmisdialogi.Action = 4
```

Ennen kuin fontinvaihtodialogin saa näkyviin, on ohjaimen **Flags**-ominaisuuteen sisällytettävä jokin seuraavista arvoista:

&H1& Ikkunassa vain näytön kirjasintyypit. (CF_SCREENFONTS)
&H2& Ikkunassa vain kirjoittimen kirjasintyypit. (CF_PRINTERFONTS)
&H3& Ikkunassa on sekä näytön että kirjoittimen kirjasintyypit (CF_BOTH)

Jos näin ei tehdä, tulee virheilmoitus, joka väittää, ettei yhtään fonttia ole asennettu. Myös muita arvoja on mahdollista asettaa, mutta vain edellä mainittu on pakollinen. Ominaisuuden voi asettaa suunnitteluvaiheessa ominaisuusikkunassa. Koodissa se on tehtävä esim. Form_Loadissa. Jos haluaa, että dialogissa näkyy mahdollisuus vaihtaa esim tekstin väriä tai alleviivata tekstiä, on Flags-asetus seuraavanlainen:

```
vakiodialogi.Flags = CF_BOTH Or CF_EFFECTS ´&H3& Or &H200&
```

Kun fontinvaihtodialogi avautuu, ei se automaattisesti näytä senhetkistä fonttia vaan se täytyy asettaa ennen dialogin näyttämistä.

```
vakiodialogi.FontName = teksti.FontName
vakiodialogi.FontSize = teksti.FontSize
vakiodialogi.Action   = 4
```

Myöskään fontin vaihtuminen uuteen ei tapahdu automaattisesti vaan sekin on koodissa asetetava.

```
teksti.FontName = vakiodialogi.FontName
teksti.FontSize = vakiodialogi.FontSize
teksti.FontItalic = vakiodialogi.FontItalic
teksti.FontBold = vakiodialogi.FontBold
teksti.FontUnderline = vakiodialogi.FontUnderline
teksti.FontStrikethru = vakiodialogi.FontStrikethru
teksti.ForeColor = vakiodialogi.Color
```

7.8.5 Tulostus (Print)

Tulostusdialogi saadaan näkyviin, kun **Action**-ominaisuuden arvoksi asetetaan viisi.

```
vakiodialogi.Action = 5
```

Tulostusdialogin keskeisiä ominaisuuksia ovat mm.

hDc	Palauttaa laiteyhteyden, jos lippu PD_RETURNDC on asetettu.
Copies	Ilmoittaa, montako kopioita tulostetaan.
FromPage	Miltä sivulta tulostus aloitetaan. PD_PAGENUMS-lippu pitää olla asetettu.
ToPage	Viimeinen tulostettava sivu. PD_PAGENUMS-lippu pitää olla asetettu.

Seuraavassa on esimerkki yksinkertaisesta tulostuksesta, jossa ei kuitenkaan käytetä tulostusdialogia, vaan tulostus menee automaattisesti oletuskirjoittimelle.

```
Dim tulostettava As String
screen.MousePointer = 11         'Kursori muutetaan tiimalasiksi.
printer.FontName = teksti.FontName
printer.FontSize = teksti.FontSize
printer.FontUnderline = teksti.FontUnderline
printer.FontBold = teksti.FontBold
printer.FontItalic = teksti.FontItalic
printer.Print teksti.Text        'Tulostetaan tekstilaatikon teksti.
printer.EndDoc                   'Lähettää tulostettavan printterille.
```

7.9 Viestidialogi (MessageBox)

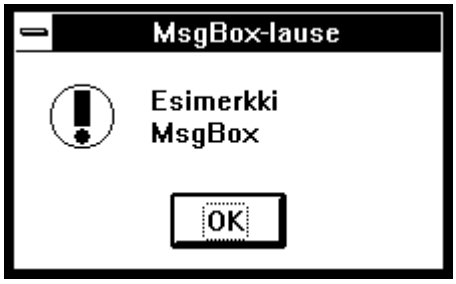
Viestidialogin avulla voi käyttäjälle näytää lyhyitä viestejä tai esittää kysymyksiä, joihin voi vastata kyllä tai ei. Dialogin koko riippuu sen sisällöstä. Dialogeja on kahta tyyppiä: **MsgBox**-funktio palauttaa käyttäjän painaman näppäimen arvon ja **MsgBox**-lauseke, joka ei palauta arvoa. Dialogien syntaksit ovat samanlaiset.

MsgBox(*viesti* **[, [***tyyppi***][,** *otsikko***]])** (funktio)

MsgBox *viesti* **[, [***tyyppi***][,** *otsikko***]]**

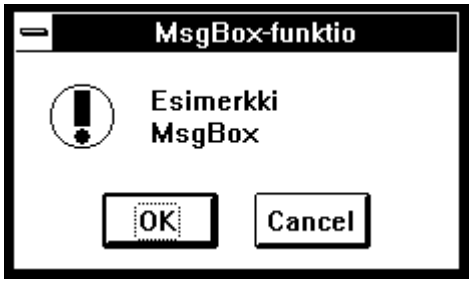
viesti	Viesti, joka näytetään dialogissa. Jos halutaan useampi rivi, pitää joka rivin loppuun lisätä carriage return ja linefeed.
tyyppi	Kokonaisluku, joka määrää esim. mikä ikoni näytetään ja mitä nappeja dialogissa on.
otsikko	Dialogin otsikko.

Esimerkki näyttää viestin ja odottaa käyttäjän kuittausta (OK-napin painaminen).




```
MsgBox "Esimerkki" & Chr(13) & Chr(10) & "MsgBox", MB_ICONEXCLAMATION,
"MsgBox-lause"
```

MsgBox-funktio palauttaa käyttäjän painaman napin. Palautusarvo on kokonaisluku. Esimerkiksi jos käyttäjä painaa OK -nappia, palautusarvo on yksi (IDOK).




```
painettu = MsgBox("Esimerkki" & Chr(13) & Chr(10) & "MsgBox", MB_OKCANCEL +
MB_ICONEXCLAMATION, "MsgBox-funktio")
```

7.10 Syöttödialogi (InputBox)

Syöttödialogi (InputBox) on vakiokokoinen modal-dialogi, jolla luetaan käyttäjän syöttämä tieto. Funktioita on kahta tyyppiä: **InputBox$** palauttaa merkkijonon ja **InputBox** Variant-tyypin. InputBoxin syntaksi on seuraava:

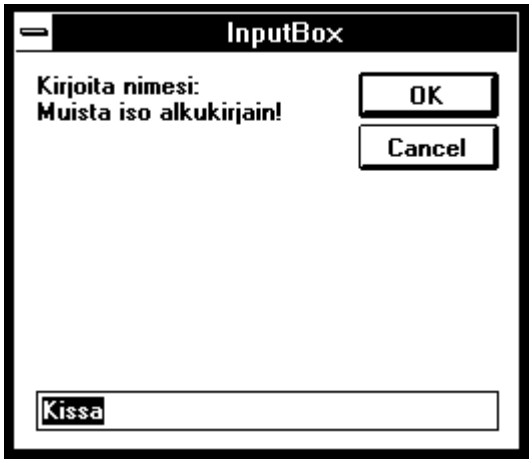
InputBox[$](*viesti* **[,** [*otsikko*] **[,** [*oletus*]**[,** *x, y* **]]]).**

viesti	Viesti, joka näytetään dialogissa. Jos halutaan useampi rivi, pitää joka rivin loppuun lisätä carriage return ja linefeed.
otsikko	Syöttödialogin otsikko.
oletus	Oletusteksti, joka näkyy dialogin syöttörivillä.
x, y	Dialogin vasen yläkulma.

Vain *viesti* on pakollinen parametri.

```
tulos = InputBox("Kirjoita nimesi:" & Chr(13) & Chr(10) & "Muista iso alku-
kirjain!", "InputBox", "Kissa")
```

Kun käyttäjä painaa OK-nappia, syöttörivin teksti sijoitetaan muuttujaan tulos. Jos painetaan Cancel-nappia, funktio palauttaa tyhjän merkkijonon.



7.11 Virheenkäsittely

Virheenkäsittelyn saa koodissa päälle kirjoittamalla lauseen **On Error** *tee jotakin*. Esimerkiksi

```
On Error GoTo virhe
...
virhe:
Select Case Err
  Case ERR_DEVICEUNVAILABLE
      MsgBox Error(Err), MB_ICONSTOP, "Virhe"
  ...
End Select
Resume
```

Virheenkäsittelyyn liittyvät funktiot **Err** ja **Error** sekä lauseet **Resume** ja **Resume Next**.

Err	Funktio palauttaa virhenumeron.
Error	Funktio palauttaa virhenumeroon liittyvän virheviestin.
Resume	Palauttaa kontrollin lauseeseen, jossa virhe syntyi. Esimerkiksi, jos korppuasema on ollut tyhjä ja siitä on tullut virhe, annetaan käyttäjälle mahdollisuus laittaa korppu asemaan ja tämän jälkeen jatketaan samasta kohdasta.
Resume Next	Jatkaa ohjelman suoritusta virhelausetta seuraavasta kohdasta.

Virheenkäsittely saadaan pois päältä lauseella **On Error GoTo 0**.

Yksi tapa käsitellä virheitä on paikallistaa mahdolliset kohdat koodissa, joissa virhe voi tapahtua (esim. tiedoston avaus tai tallettaminen jne.) ja laittaa virheenkäsittely päälle vain tämän lauseen (lauseiden) ajaksi.

```
Sub levyasema_Change ()                'virhe on lomaketason muuttuja.
  On Error Resume Next                'Jos virhe, jatka seuraavasta.
  hakemisto.Path = levyasema.Drive    'Päivitetään hakemistolistaa.
  virhe = Err                         'Otetaan virhenumero talteen.
  On Error GoTo 0                     'Virheenkäsittely pois päältä, Err = 0.
  virheenkasittely virhe              'Kutsutaan virheenkäsittelyohjelmaa.
End Sub                               'Parametrina tullut virhe tai 0, jos ei
                                       'tullut virhettä.
```

7.12 Avustustiedostot

VB-ohjelmiin voidaan liittää avustustiedosto, joka kirjoitetaan jollakin **rtf**-tekstimuodon tunnistavalla tekstinkäsittelyohjelmalla (esim. Microsoft Word). VB:ssa toimitetaan mukana **Help**-kääntäjä ja **Hot Spot** -editori, jolla voi luoda linkkejä kuviin.

Valikkokohdassa **Options - Project** voidaan määritellä ohjelmaan liittyvän avustustiedoston nimi. Aina kun käyttäjä painaa ohjelmassa **F1**, ilmestyy ruudulle avustustiedosto, jos sellainen on. Mikäli lomakkeen ohjaimiin on liitetty **HelpContextID**-numero, avautuu avustustiedosto määrätystä kohtaa. Muussa tapauksessa näkyviin tulee avustustiedoston ensimmäinen sivu. HelpContextID on kokonaisluku ja sitä vastaava

avustustiedoston linkki määritellään **.HPJ**-tiedoston **[MAP]**-osassa. (Esimerkkiohjelma HELP.MAK, KOEHELP.HPJ, KOEHELP.HLP, KOEHELP.RTF)

```
[OPTIONS]
TITLE=Koehelppi
COMPRESS=false
[FILES]
koehelp.rtf
[MAP]
EKALINKKI 1
TOKALINKKI 2
KOLMASLINKKI 3
```

Huom! Isoilla ja pienillä kirjaimilla on ero linkkien nimissä.

Avustustiedoston kutsuminen voidaan hoitaa myös ilman F1-näppäintä. Esimerkissä (HELPDLL.MAK, projektiin HELPDLL.FRM ja HELPDLL.BAS) avustustiedostoa kutsutaan painamalla ohjelman Apua-nappia.

7.13 MDI (Multiple-Document Interface)

MDI-sovellus sallii lomakkeen usean kopion näyttämisen yhtäaikaa. Muun muassa useat tekstinkäsittelyohjelmat ovat MDI-sovelluksia. Sovelluksella on yksi isä-ikkuna, jonka raamien sisällä lapsi-ikkunat näytetään. Lapsi-ikkunoita ei voi vetää isä-ikkunan raamien ulkopuolelle.

Visual Basic -ohjelmassa vain yksi lomake voi olla MDI-lomake. Lapsi-ikkuna on tavallinen lomake, jonka **MDIChild**-ominaisuus on asetettu todeksi.

1. Luo MDI-lomake (isä-ikkuna). Valitse **File - New MDI Form**.
2. Luo lapsi-ikkuna(t). Valitse **File - New Form** tai avaa jo valmis lomake ja aseta lomakkeen **MDIChild = True**.

Uusia lapsi-ikkunoita saadaan, kun valmiista lapsi-ikkunasta luodaan uusia esiintymiä (kopioita). Jos lapsi-ikkunan nimi on *lapsi*, niin uusi esiintymä luodaan

```
Dim uusilapsi As New lapsi
uusilapsi.Show
```

Lapsi-ikkunat jakavat saman koodin, mutta jokainen lapsi-ikkuna tunnistaa omat tapahtumansa. Kun koodista jätetään pois viittaukset lomakkeeseen, se vaikuttaa aina siihen lomakkeeseen, joka sitä kulloinkin kutsuu, esim.

```
Form1!teksti.Height = ScaleHeight 'Vaikuttaa Form1:n tekstilaatikkoon.
teksti.Height = ScaleHeight 'Vaikuttaa kutsuvan lomakkeen tekstilaatikkoon.
```

Toinen tapa viitata sillä hetkellä aktiiviseen lapsi-ikkunaan on avainsana **Me** tai **ActiveForm**.

Jos ja kun ohjelmassa halutaan pitää kirjaa siitä, onko lapsi-ikkunan sisältö talletettu vai ei, voidaan se tehdä esimerkiksi pitämällä yllä taulukoita lapsi-ikkunoista, talletustilasta ja avatuista tiedostoista.

```
'moduuli – declarations
```

```
Type Tila
   talletettu As Integer
   tiedostonimi As String
End Type

Global lomakkeet() As New editori 'editori on lapsilomake.
Global talletustila() As Tila
```

Joka kerran kun käyttäjä avaa uuden ikkunan tai lataa tiedoston, globaaleja taulukoita kasvatetaan yhdellä. Uuden lapsi-ikkunan **Tag**-ominaisuuteen talletetaan lomakkeen indeksi taulukossa.

```
Dim laskuri As Integer
'alustus
ReDim talletustila(0)                    'Luodaan taulukon 1. alkio.
ReDim lomakkeet(0)
lomakkeet(0).Tag = 0
talletustila(0).talletettu = True
talletustila(0).tiedostonimi = ""

'uusi
laskuri = UBound(lomakkeet)
ReDim Preserve lomakkeet(laskuri + 1)
ReDim Preserve talletustila(laskuri + 1)
lomakkeet(laskuri + 1).Tag = laskuri + 1
talletustila(laskuri + 1).talletettu = True
talletustila(laskuri + 1).tiedostonimi = ""
lomakkeet(laskuri + 1).Show

'avaa
'Sama kuin kohta uusi, mutta asetetaan tiedoston nimi talteen
'ja ikkunan otsikkoon.
'kasiteltava_tiedosto on avatun tiedoston nimi.
talletustila(laskuri + 1).tiedostonimi = kasiteltava_tiedosto
lomakkeet(laskuri + 1).Caption = "Tekstieditori -" & kasiteltava_tiedosto
```

Joka kerran, kun lapsi-ikkuna suljetaan, aktivoituu tapahtuma **QueryUnload**. Jos itse isäikkuna suljetaan lähetetään QueryUnload-tapahtuma jokaiselle lapsi-ikkunalle. Tämän tapahtuman koodissa on hyvä testata, onko ikkunan sisältö talletettu vai ei.

```
Sub Form_QueryUnload (cancel As Integer, UnloadMode As Integer)
   Dim message As String, vastaus As Integer
   If talletustila(Me.Tag).talletettu = False Then
      message = "Talletetaanko?" & Chr$(13) & Chr$(10) & Me.Caption
      vastaus = MsgBox(message, 36, "Varmistus")
      Select Case vastaus
         Case IDYES
            'talleta
         Case IDNO
            'ei talleteta
      End Select
   End If
End Sub
```

Isä- ja lapsi-ikkunaan voidaan kumpaankin liittää erillinen valikko (Menu). Isäikkunan valikko näky silloin, kun yhtään lapsi-ikkunaa ei ole ladattuna. Kun lapsi-ikkuna ladataan, siirtyy lapsi-ikkunaan liitetty valikko näkymään isä-ikkunassa.

Lapsi-ikkunoiden esittämistä voidaan säädellä vakioilla TILE_HORIZONTAL ja CASCADE (löytyvät tiedostosta CONSTANT.TXT). Luodaan lapsi-ikkunalle Ikkuna-valikko ja liitetään siihen tarvittava koodi.

```
Sub menu_cascade_Click ()
```

```
  editorimdi.Arrange CASCADE     'editorimdi on isäikkuna.
End Sub

Sub menu_tile_Click ()
   editorimdi.Arrange TILE_HORIZONTAL
End Sub
```

Tämän saman esimerkin olisi voinut tehdä myös siten, että Tag-ominaisuuteen olisi talletettu sekä talletustila että tiedoston nimi samaan merkkijonoon (esim. "Etiedostonnimi"). Merkkijonon ensimmäinen merkki olisi kertonut tiedoston talletustilan (E / T). Tämä olisi kuitenkin huono ratkaisu silloin, jos tietoja tarvittaisiin enemmän.

7.14 DLL (Dynamic-link libraries)

Ohjelmassa käytettävät DLL-kirjastojen aliohjelmat esitellään joko lomakkeen tai moduulin declararion-osassa. Esittely tapahtuu **Declare**-lauseen avulla.

Declare Sub|Function *nimi* **Lib** *"DLL-kirjaston nimi"* **Alias** *"nimi"*(*argumenttilista*) *palautustyyppi*

Sub	Jos aliohjelma ei palauta arvoa.
Function	Jos aliohjelma palauttaa arvon.
nimi	Kutsuttavan aliohjelman nimi.
DLL-kirjaston nimi	Merkkijono, joka on kirjaston nimi. Ei polkua eikä tarkennetta, jos ei kyseessä oma kirjasto.
nimi	Aliaksen avulla voidaan kutsuttava aliohjelma nimetä uudelleen. Esim. VB ei salli alaviivalla alkavia nimiä ja siten esim. funktio *_lopen* on nimettävä uudelleen aliaksen avulla.

```
Declare Function LOpen Lib "Kernel" Alias "_lopen" (ByVal fn As String,
ByVal f As Integer) As Integer
```

argumenttilista	Kutsuttavan aliohjelman vaatimat parametrit.
palautustyyppi	Palautustyyppi, jos kyseessä funktio.

Argumenttilistassa voidaan käytää sanaa **ByVal** samassa merkityksessä kuin VB:n aliohjelmissa. Poikkeuksen kuitenkin tekevät merkkijonot. Merkkijonojen yhteydessä ByVal ei merkitse, että parametri välitettäisiin arvona viittauksen sijasta, vaan että se käännetään **NUL**-merkkiin loppuvaksi merkkijonoksi.

7.14.1 Valmiiden funktioiden kutsuminen

Tämä pieni esimerkki (DLLINI.MAK) lukee otsikon (Label) fontin INI-tiedostosta ja tallettaa sinne uuden fontin. Kumpikin funktio esitellään lomakkeen declarations-osassa:

Huom! ByVal merkkijonojen yhteydessä lisää NUL-merkin merkkijonon loppuun.

```
Declare Function WritePrivateProfileString Lib "Kernel" (ByVal Section As
String, ByVal Item As String, ByVal st As String, ByVal file As String)
As Integer
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal Section As
String, ByVal Item As String, ByVal default As String, ByVal destbuff As
String, ByVal destbufflen As Integer, ByVal file As String) As Integer
```

```
Dim fonttinimi As String * 20
```

Otsikon fontti luetaan Form_Load:issa.

```
tulos = GetPrivateProfileString("FONTTI", "FonttiNimi", "Arial", fonttinimi,
Len(fonttinimi), ".\koe.ini")
```

Uusi fontti kirjoitetaan INI:in.

```
uusifontti = InputBox("Anna uusi fontti:")
tulos = WritePrivateProfileString("FONTTI", "FonttiNimi", uusifontti,
".\koe.ini")
```

7.14.2 Itsetehtyjen aliohjelmien kutsuminen

Tässä esimerkissä tehdään Borlandin C++:lla ensin DLL-kirjasto, jossa on aliohjelma karkausvuosien selvittämiseksi. Sen jälkeen sitä aliohjelmaa käytetään VB-ohjelmassa.

DLL-kirjaston luomisen vaiheet:

1. Tee pääohjelma PVMDLL.C, joka sisältää halutut aliohjelmat.

```
/**************/
/* pvmdll.c   */
/*****************************************************************************/

  MODULE:  pvmdll.c
  PURPOSE: Esimerkki yksinkertaisesta DLL-kirjastosta.
  AUTHOR:  Vesa Lappalainen 28.8.1994
  UPADTE:
  USAGE:   Kirjaston käyttämiseksi
       1) Lisätään omaan ohjelmaan:
            #include "pvmlib.h"
       2) Kutsut aliohjelmiin:
            if ( karkausvuosi(1994) ) ...
       3) Lisätään omaan .DEF-tiedostoon:
            IMPORTS PVMDLL.karkausvuosi
          tai
            lisätään projektiin .LIB, johon tarvittavat .DLLät on lisätty
       4) Varmistetaan, että ajon aikana PVMDLL.DLL on joko oletushakemis-
                  tossa tai polussa.

       PS.  Tämä tiedoston pääohjelma toimii melkein sellaisenaan minkä
            tahansa DLL:än "pääohjelmana".
/*****************************************************************************/
#include <windowsx.h>
#include "pvmlib.h"
/*****************************************************************************/
int CALLBACK _export karkausvuosi(int vuosi)
{
  if ( vuosi % 100 ) return (vuosi % 4 ) == 0;
  return ( vuosi % 400 ) == 0;
}
```

2. Kirjoita PVMLIB.H-tiedosto.

```
#ifdef __DLL__
#define WINAPI_LIB WINAPI _export
#else /* ei DLL */
#define WINAPI_LIB WINAPI
#endif
int WINAPI_LIB karkausvuosi(int vuosi);
```

3. Tee PVMDLL.DEF-tiedosto.

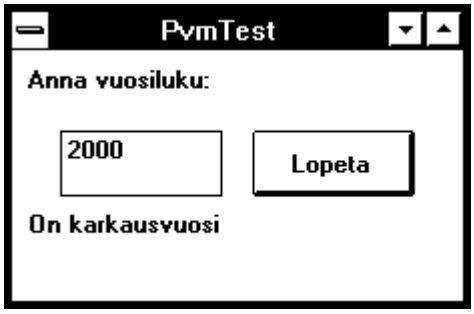
```
LIBRARY       PVMDLL                    ;application's module name
DESCRIPTION  'Tekee paivamaarahommia'
EXETYPE       WINDOWS                   ;required for all Windows applica-
tions
PROTMODE
CODE  PRELOAD MOVEABLE DISCARDABLE
DATA  PRELOAD MOVEABLE SINGLE           ;HUOM! SINGLE
HEAPSIZE     1024
```

4. Kirjoita pääohjelma PAADLL.C.

```
#include <windowsx.h>

HINSTANCE DLLG_hInstance; /* Globaalit muuttujat, joita tarvitaan. */

int CALLBACK _export WEP(int nParam)
{
#pragma argsused
  return 1;
}

/*********************************************************************/
int WINAPI LibMain(HINSTANCE hInstance, WORD wDataSeg, WORD wHeapSize, LPSTR
lpCmdLine)
{
#pragma argsused
  if ( wHeapSize > 0 ) UnlockData(0);
  return 1;
}
```

4. Tee seuraavaksi projekti, jossa tiedostot PVMDLL.C, PVMDLL.DEF ja PAADLL.C ja käännä DLL-kirjastoksi. (**Huom!** Muista laittaa käännöksen tulokseksi Windows DLL.)

VB:ssa tehdään ohjelma, joka kertoo, onko käyttäjän syöttämä vuosiluku karkausvuosi vai ei.



Kirjastofunktio esitellään lomakkeen declarations-osassa.

```
Declare Function karkausvuosi Lib "PvmDLL.DLL" (ByVal vuosi As Integer)
As Integer
```

```
Sub vuosi_Change ()
  If (Not IsNumeric(vuosi)) Then
    kv = "Ei numero!"
  ElseIf (karkausvuosi(vuosi) = 0) Then
    kv = "Ei ole karkausvuosi"
  Else
    kv = "On karkausvuosi"
  End If
End Sub
```

DLL-kirjaston täytyy sijaita oletushakemistossa tai polussa olevassa hakemistossa.

8. Tiedonsiirto muiden ohjelmien kanssa

8.1 Tiedostot

Tiedostot avataan käskyllä Open:

Open *file* [**For** *mode*] [**Access** *access*] [*lock*] **As** [#]*filenumber* [**Len** = *reclen*]

file	Tiedoston nimi + mahdollinen polku.
mode	Tiedoston tyyppi: **Append, Binary, Input, Output, Random**.
access	Sallitut operaatiot: **Read, Write, Read Write**
lock	Sallitut operaatiot muille avatun tiedoston käyttäjille: **Shared, Lock Read, Lock Write, Lock Read Write**
filenumber	Tiedostokahva
reclen	Tietueen pituus (random) tai luettavat merkit (sequential).

Tiedostot suljetaan lauseella **Close** *filenumber*.

8.1.1 Suorasaantitiedostot (Random)

Suorasaantia voidaan käyttää esimerkiksi tietueiden tallettamiseen. Suorasaantitiedosto avataan käskyllä Open:

Open *tiedostonimi* **For Random As** *tiedostonro* **Len** = *pituus*

```
Type ihminen
  nimi As String * 20     'Merkkijonon oltava vakiopituinen, jos halutaan
  paino As Double         'tallentaa tiedostoon.
  pituus As Double
End Type
```

```
Dim nro As Integer, iikka As ihminen
Dim pituus As Long
pituus = Len(iikka)
nro = FreeFile
Open "koe.ran" For Random As #nro Len = pituus
```

Tiedostosta luetaan lauseella **Get**.

Get *tiedostonro, [kohta], tietue*

tiedostonro	Jo auki olevan tiedoston numero.
kohta	Se kohta tiedostosta, mistä luetaan. Jos kohtaa ei anneta, luetaan nykyisestä kohdasta (= **Seek**(tiedostonro) ks. alla)
tietue	Muuttuja, johon tiedot luetaan.

```
Get #nro, 1, iikka  'Lukee muuttujaan iikka tiedoston kohdasta yksi.
                    'Siirtää osoittimen seuraavaan kohtaan.
```

Tiedostoon kirjoitetaan lauseella **Put**.

Put #*tiedostonro, [kohta], tietue*

tiedostonro	Jo auki olevan tiedoston numero. Huom. risuaita!
kohta	Se kohta tiedostosta, johon kirjoitetaan. Jos kohtaa ei anneta, kirjoitetaan nykyiseen kohtaan
tietue	Muuttuja, josta tiedot luetaan.

```
Put #nro, 1, iikka 'Kirjoitetaan uusi tieto tiedoston kohtaan 1 (alkuun).
                   'Siirtää osoittimen seuraavaan kohtaan.
```

Seek (*tiedostonro***)**	Palauttaa sen tiedoston kohdan, johon tiedosto-osoitin sillä hetkellä osoittaa.
Seek *tiedostonro, kohta*	Asettaa tiedosto-osoittimen osoittamaan tiedoston kohtaa, mistä seuraavaksi luetaan tai mihin seuraavaksi kirjoitetaan.

```
Seek #nro, LOF(nro)\Len(iikka)+1 'Asetetaan osoitin tiedoston loppuun.
Put #nro, , iikka                'Kirjoitetaan uusi tieto osoittimen koh-
taan.
```

8.1.2 Peräkkäistiedostot (Sequential)

Peräkkäistiedostoja voi käyttää esimerkiksi tekstitiedostojen käsittelyyn. Tiedosto avataan käskyllä Open:

Open *tiedostonimi* **For Input As** *tiedostonro*

```
Dim rompe As String, pituus As Long, tiedostokahva As Integer
tiedostokahva = FreeFile  'Pyydetään tiedostokahva tiedoston avaamiseen.
pituus = FileLen(nimi)    'Tarkistetaan tiedoston pituus ennen avaamista.
If pituus >= 60000 Then   'Jos pituus enemmän kuin tekstilaatikkoon mahtuu.
  MsgBox "Liian iso tiedosto avattavaksi.", 16, "Avaa..."
  Exit Function
End If
                          'Avataan tiedosto.
Open "koe.txt" For Input As tiedostokahva
                          'Luetaan koko sisältö merkkijonoon.
rompe = Input$(LOF(tiedostokahva), tiedostokahva)
                                'LOF palauttaa avatun tiedoston koon.
Close tiedostokahva 'suljetaan tiedosto
```

Tiedostoon kirjoitetaan käskyllä **Print**. Talletus korvaa tiedoston entisen sisällön.

```
tiedostokahva = FreeFile
Open "koe.txt" For Output As tiedostokahva
'Kirjoitetaan tekstilaatikon sisältö tiedostoon.
Print #tiedostokahva, teksit.Text
Close tiedostokahva 'suljetaan tiedosto
```

Jos halutaan kirjoittaa tiedoston loppuun ja säilyttää tiedostossa jo oleva tieto, käytetään avattaessa sanaa **Append**, joka asettaa osoittimen tiedoston loppuun.

```
tiedostokahva = FreeFile
Open "koe.txt" For Append As tiedostokahva
'Kirjoittaa tekstilaatikon sisällön tiedostoon edellisten tietojen perään.
Print #tiedostokahva, teksti.Text
Close tiedostokahva
```

8.1.3 Binääritiedostot (Binary)

Binääritiedosto avataan käskyllä Open:

Open *tiedostonimi* **For Binary Access Read As** *tiedostonro*

```
tiedostokahva = FreeFile
'Avataan tiedosto luettavaksi.
Open "koe.bin" For Binary Access Read As tiedostokahva
```

Tiedostosta luetaan lauseella **Get**.

Get #*tiedostonro, [kohta], muuttuja*

tiedostonro	Kahva avattuun tiedostoon.
kohta	Mistä kohtaa tiedostoa luetaan. Jos kohtaa ei anneta, luetaan nykyisestä kohdasta.
muuttuja	Muuttuja, johon tiedot luetaan.

```
Dim rompe As String * 1000
If Not EOF(tiedostonro) Then
 'Luetaan tiedostosta 1000 tavua merkkijonoon.
  Get #tiedostonro, 1, rompe
End If
```

Tiedostoon kirjoitetaan lauseella **Put**.

Put #*tiedostonro, [kohta], muuttuja*

tiedostonro	Kahva avattuun tiedostoon.
kohta	Mihin kohtaan tiedostoa kirjoitetaan. Jos kohtaa ei anneta, kirjoitetaan nykyiseen kohtaan.
muuttuja	Muuttuja, josta tiedot luetaan.

8.2 Leikepöytä

Esimerkkiohjelma (LEIKE.MAK) siirtää tekstiä tai kuvia leikepöydälle tai liittää leikepöydältä kuvia tai tekstiä. Ohjelmassa on kuvakehys ja tekstilaatikko, joten aina ensin testataan, kumpi niistä on aktiivinen **Screen.ActiveControl** ja sen mukaan päätetään, käsitelläänkö kuvaa vai tekstiä.

Kopioi (Copy):

```
clipboard.Clear
If TypeOf Screen.ActiveControl Is TextBox Then
  'Laitetaan leikepöydälle maalattu teksti.
   clipboard.SetText Screen.ActiveControl.SelText
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
  'Kun kuvan tyyppi on 0, VB päättää itse tyypin.
   clipboard.SetData Screen.ActiveControl.Picture, 0
End If
```

Leikkaa (Cut):

```
If TypeOf Screen.ActiveControl Is TextBox Then
```

```
  clipboard.SetText Screen.ActiveControl.SelText
  Screen.ActiveControl.SelText = ""
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
  clipboard.SetData Screen.ActiveControl.Picture, 0
  Screen.ActiveControl.Picture = LoadPicture()
End If
```

Liitä (Paste):

```
If TypeOf Screen.ActiveControl Is TextBox Then
  'Leikepöydältä tuotu teksti korvaa tekstilaatikon
  'maalatun tekstin tai jos mitään ei ole maalattu,
  'sijoittaa leikepöydältä tuodun tekstin kursorin
  'kohtaan.
  Screen.ActiveControl.SelText = clipboard.GetText()
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
  Screen.ActiveControl.Picture = clipboard.GetData()
End If
```

8.2.1 Tekstin siirtäminen

Tekstiä siirretään leikepöydälle metodilla **SetText**.

Clipboard.SetText *Teksti [,Format].*

Teksti	Sisältää siirrettävän tekstin.
Format	Ilmaisee siirrettävän tekstin muodon. CF_LINK (DDE-linkin tiedot) CF_TEXT (teksti)

Jos muotoa ei anneta, CF_TEXT on oletus.

```
clipboard.SetText tekstilaatikko.Text          'Siirtää leikepöydälle
                                               'tekstilaatikon kaikki
                                               'tekstin.
clipboard.SetText tekstilaatikko.SelText       'Siirtää leikepöydälle vain
                                               'valitun tekstin.
```

Tekstiä liitetään (Paste) leikepöydältä metodilla **GetText**.

Clipboard.GetText(*Format***)**.

Format on liitettävän tekstin muoto ja oletuksena se on CF_TEXT.

```
tekstilaatikko.Text = clipboard.GetText()      'Leikepöydältä tuotu teksti
                                                      'korvaa tekstilaatikon
                                                             'entisen teks-
tin.
tekstilaatikko.SelText = clipboard.GetText()   'Sijoitetaan leikepöydältä
                                                      'tuotu teksti kursorin
                                                             'kohtaan.
```

Jos tekstilaatikossa olevasta tekstistä maalataan osa, korvaa leikepöydätä siirrettävä teksti maalatun tekstin. Jos halutaan vain lisätä tekstiä kursorin osoittamaan kohtaan, ei maalattuna saa olla mitään.

8.2.2 Kuvien siirtäminen

Kuvia siirretään leikepöydälle metodilla **SetData**.

Clipboard.SetData *Data [,Format].*

Data on kuva, joka halutaan kopioida leikepöydälle. Format on jokin seuraavista:

CF_BITMAP	kuva.bmp
CF_METAFILE	kuva.wmf
CF_DIB	device-independent bmp
CF_PALETTE	väripaletti

Jos Format on nolla, VB päättelee automaattisesti kuvan tyypin.

```
clipboard.SetData kuva.Picture, 0
```

Kuva liitetään leikepöydältä (Paste) esimerkiksi kuvakehykseen metodilla **GetData**.

Clipboard.GetData(*Format*).

Format on liitettävän kuvan tyyppi. Jos format on nolla tai sitä ei anneta, VB päättää automaattisesti kuvan tyypin.

```
kuva.Picture = clipboard.GetData()
```

Metodilla **GetFormat** on mahdollista tutkia leikepöydän sisältöä.

Clipboard.GetFormat(*Format*).

Format voi olla jokin edellä mainituista. GetFormat palauttaa arvon tosi, jos leikepöydältä löytyy kysytyn tyyppistä tietoa, muuten palautetaan epätosi.

8.3 DDE (Dynamic Data Exchange)

8.3.1 Linkit ilman koodia (Paste Link)

Esimerkkinä luodaan vastaanottava linkki Visual Basicin (vastaanottaja) ja Quattro Pron (lähettäjä) välille:

- Valitaan Quattrosta kenno (kennot), jotka halutaan linkittää.
- Valitaan valikosta Copy.
- Siirrytään VB:hen ja aktivoidaan lomakkeen tekstilaatikko.
- Valitaan valikosta Paste Link.

Nyt on luotu automaattinen, pysyvä linkki, joka voidaan tallettaa ohjelman mukana. Linkki vaatii molempien ohjelmien käynnissä olemisen.

(Tekstilaatikon **LinkTopic**-arvo on tyyppiä
"QPW|D:\TEKSTEJA\QPRO\NOTEBK1.WB1")

Luodaan lähettävä linkki VB:n (lähettäjä) ja Quattron (vastaanottaja) välille.

- Aseta VB:ssä lomakkeen LinkMode-ominaisuuden arvoksi 1 - Source.
- Aktivoi lähettävä ohjain.
- Valitse valikosta Copy ja siirry Quattroon.
- Quattrossa valitse kenno ja valikosta Paste Link.

(Kennon linkki on esim. @DDELINK([PROJECT1|FORM1]"TEKSTI").)

8.3.2 Linkit koodista

DDE-linkkeihin liitttyvät oleellisesti ominaisuudet **LinkTopic, LinkItem** ja **LinkMode**.

LinkTopic	on muotoa *application*|*topic* esim. (WINWORD|D:\TEKSTEJA\ESIM.DOC).
Application	on sovelluksen nimi esim. WINWORD, EXEL, QPW. Visual Basicissa se on EXE-nimi tai projektin nimi - kumpikin ilman tarkennetta.
Topic	on keskustelun aihe ja se on usein tiedoston nimi polkuineen, esim. D:\WINWORD\ESIM.DOC.
LinkItem	on keskustelun konkreettinen asia esim. Exelissä kenno R1C1 tai WinWordissa \DOC eli mikä tahansa tiedosto.
LinkMode	on linkin laatu ja sillä voi olla arvot:
0	ei linkkiä
1	**automaattinen** eli vastaanottavassa ohjaimessa tieto päivittyy aina, kun lähettävässä päässä tieto muuttuu.
2	**manuaalinen** eli vastaanottava pää päivittyy vain metodilla LinkRequest.
3	**Notify** Kun tieto muuttuu, syntyy LinkNotify-tapahtuma ja päivitys tapahtuu vain metodilla LinkRequest.

Alla olevassa esimerkissä luodaan vastaanottava automaattinen linkki VB-ohjelman ja Wordin välille. Ohjelman tekstilaatikko päivittyy aina, kun sitä klikataan.

```
Sub teksti_Click ()
  teksti.LinkTopic = "WINWORD|D:\OHJELMOI\VISUALB\OPETUS\DDE.DOC"
  teksti.LinkItem = "\DOC"
  teksti.LinkMode = AUTOMATIC
End Sub
```

Luodaan seuraavaksi vastaanottava manuaalinen linkki VB-ohjelman ja Quattro Pron välille. Ohjelman tekstilaatikko päivittyy aina, kun sitä klikataan.

```
Sub teksti_Click ()
  Const MANUAL = 2, NONE = 0
  If teksti.LinkMode = NONE Then
    teksti.LinkTopic = "QPW|D:\TEKSTEJA\QPRO\NOTEBK1.WB1"
    teksti.LinkItem = "$A:$A$1..$A:$A$1"
    teksti.LinkMode = MANUAL
  End If
  teksti.LinkRequest
End Sub
```

Seuraavaksi luodaan vastaanottava Notify-linkki VB-ohjelman ja Quattro Pron välille. Tekstilaatikko päivittyy automaattisesti, jos LinkRequest laitetaan LinkNotify-tapahtuman koodiin..

```
Sub Form_Load ()
  Const NOTIFY = 3, NONE = 0
  If teksti.LinkMode = NONE Then
    teksti.LinkTopic = "QPW|D:\TEKSTEJA\QPRO\NOTEBK1.WB1"
    teksti.LinkItem = "$A:$A$1..$A:$A$1"
    teksti.LinkMode = NOTIFY
  End If
End Sub
```

```
Sub teksti_LinkNotify ()
  teksti.LinkRequest
End Sub
```

Manuaalisella ja Notify-linkillä voidaan kontrolloida linkkiä ja esimerkiksi määrätä, koska halutaan päivittää tietoja.

Luodaan lähettävä linkki VB:n (lähettäjä) ja Quattro Pron (vastaanottaja) välille.

- VB:ssä lomake.LinkMode = 1 (Source)
- VB:ssä valitaan ohjain, joka on lähettäjä (esim tekstilaatikko)
- Quattrossa kirjoitetaan haluttuun kennoon linkki, joka on esimerkiksi @DDELINK([PROJECT1|FORM1]"TEXT1")

Metodia **LinkSend** voi käyttää vain kuvien kanssa.

- VB:ssä: lomake.LinkMode = 1 (Source)
- Form_Loadiin koodi `kuva.LinkSend`.
- WORD:ssä valikosta **Insert Field**, teksti: **ddeauto project1 form1 picture1**.

Syntyy automaattinen linkki. Kun kuva VB:ssä muuttuu, se päivittyy Wordiin.

8.3.3 Ohjelmaan Liitä linkki -valikkokomento (Paste Link)

VB-ohjelmaan voidaan tehdä valikkokomento Liitä linkki. Tällöin kopioidaan leikepöydällä mahdollisesti oleva linkki omaan ohjelmaan. Esimerkissä luodaan automaattinen vastaanottava linkki.

```
Sub menu_liita_linkki_Click ()
  Dim link As Variant, item As Integer
  link = clipboard.GetText(CF_LINK)     'Kopioidaan linkki leikepöydältä.
  'Monesko merkki "!" on merkkijonossa.
  item = InStr(link, "!")
  If item <> 0 Then
    screen.ActiveControl.LinkMode = 0
    'Erotetaan merkkijonosta LinkTopic ja LinkItem.
    screen.ActiveControl.LinkTopic = Left(link, item - 1)
    screen.ActiveControl.LinkItem = Mid(link, item + 1)
    screen.ActiveControl.LinkMode = 1
  ElseIf InStr(link, "|") Then
    screen.ActiveControl.LinkMode = 0
    screen.ActiveControl.LinkTopic = link
    screen.ActiveControl.LinkItem = ""
    screen.ActiveControl.LinkMode = 1
  End If
End Sub
```

9. Tietokanta

Visual Basic 3.0:lla voi luoda tietokantoja ja käyttää jo olemassaolevia. Luodut tietokannat ovat **Microsoft Access** -tyyppisiä ja ne voi tehdä esim. **DataManagerilla**, joka löytyy Window-valikosta.

Tietokantaohjaimen **DataBaseName-**ominaisuus on tietokannan nimi polkuineen, jos kyseessä Microsoft Access -tietokanta. Muussa tapauksessa se on vain polku hakemistoon, jossa tietokanta sijaitsee. Esimerkiksi **Paradox-**tietokanta täytyy olla talletettu 3.X muotoon eli Windows Paradox -tietokanta ei suoraan kelpaa.

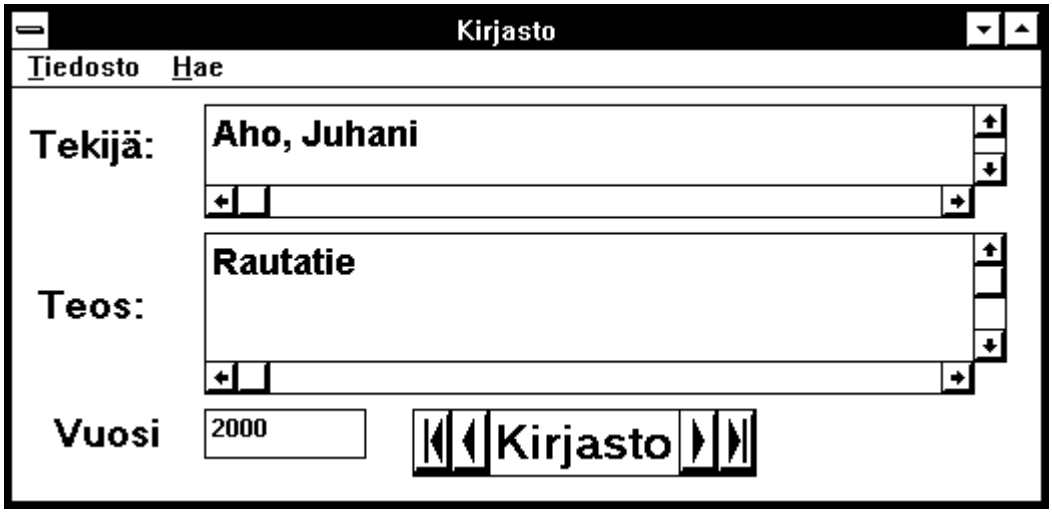
Connect-ominaisuuteen laitetaan tietokannan tyyppi (jätetään tyhjäksi, jos Access-tietokanta) esim. "paradox 3.x" tai "dbase IV".

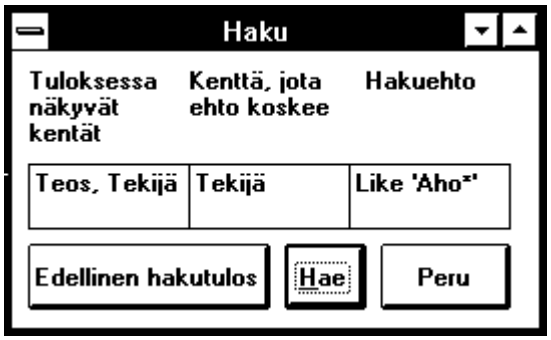
RecordSource-ominaisuuteen määritellään, mistä taulusta tietoja luetaan (painetaan ominaisuusikkunassa nuolta, jolloin saadaan näkyviin tarjolla olevat taulut).

Tämän jälkeen piirretään lomakkeelle haluttuja kenttiä varten tekstilaatikot. Kentät näihin määritellään ominaisuusikkunan avulla.

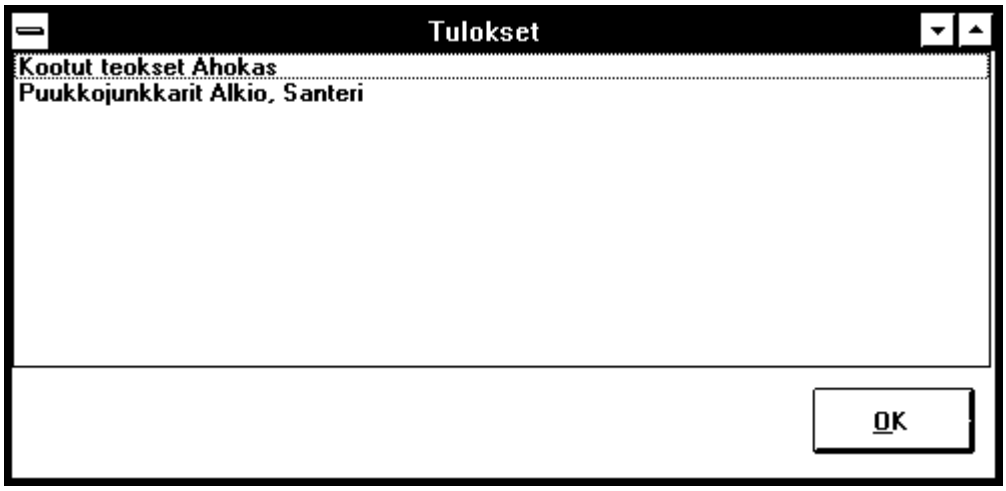
Ensin asetetaan arvo ominaisuudelle **DataSource**, joka on se nimi, joka annettiin tietokantaoliolle. Ominaisuuteen **DataField** valitaan näkymään haluttu kenttä taulusta. Tekstilaatikoita piirretään niin monta kuin kenttiä halutaan näkyviin. Kun ohjelma ajetaan, voidaan tietokantaa selata tai muuttaa tietoja eikä ole tarvinnut kirjoittaa riviäkään koodia.

Jos ja kun tietokannalle halutaan monipuolisempia toimintoja, täytyy jo kirjoittaa koodia. Tietokantakieli on **SQL**-yhteensopivaa. Esimerkkiohjelma (Projektiin KIR.FRM, KIRHAKU.FRM, KIRTUL.FRM) käyttää **Paradox 3.5** -tyyppistä kirjatietokantaa ja hakee sieltä annetuilla ehdoilla kirjoja. Hakutulos näytetään omassa ikkunassa.





Haun tulos näkyy listassa omalla lomakkeella.




```
Sub hae_Click ()

'Dynaset-tyyppiset oliot ovat tietokannan osajoukkoja.
'|---------|----------|----------|--
'| Kalle   | 33490    | Lahti    |
'|---------|----------|----------|--
'| Kaisa   | 32189    | Turku    |
'|---------|----------|----------|--
'|         |          |          |
'
'Jos tietokannasta on haettu esim. kaikki K:lla alkavat nimet (hakuehto)
'niin tulostaulu (Dynaset) on ylläolevan kaltainen. Siinä näkyvät kaikki
'tietokannan kentät.

    Dim MunKanta As Database     'tietokantaolio
    Dim MunHaku As Dynaset       '"haun tulos"
    Dim i As Integer
    Dim Rivi As String           'Rivi-merkkijonoon kootaan tulostaulusta
                                 'halutut kentät (määrätty hakuavaimessa)
                                 'rivi kerrallaan. Esim. Rivi = "Kalle
      Lahti"

    'Avataan tietokanta.
    '1. param. tietokannan nimi + polku, jos Access-tietokanta muuten polku.
    '2. param. jaetaanko tietokantaa muiden käyttäjien kanssa.
    '3. param. ReadOnly
    '4. param. Connect eli tietokannan tyyppi, jos muu kuin Access.

         Set MunKanta = OpenDatabase(TIETOKANTA, True, False, TYYPPI)

    'Luodaan osajoukko tietokannasta annetun hakuavaimen perusteella.
    'HakuAvain SQL
    'syntaksi:
    'select
```

```
    '(tuloslistassa näkyvät kentät, * = kaikki)
    'from
    '(taulun/tietokannan nimi ilman tarkenninta)
    'where
    '(kenttä, jota ehto (hakuehto) koskee)
    '
    'Halutaan hakea esim.        hakuehto
    'Kivi*                       kenttä) like 'Kivi*'
    '< 1900                      kenttä) < 1900
    '1980 < vuosi < 1990         vuosi > 1980 and vuosi < 1990
    '= 1900                      kenttä)= 1900
    '= Kivi, Aleksis             kenttä)= 'Kivi, Aleksis'

    On Error GoTo virhe
    Set MunHaku = MunKanta.CreateDynaset("select " & kentät.Text & " "
    & "from " & TIETOKANNANNIMI & "where " & ehtokenttä.Text & " "
    & ehto.Text)

    tulosikkuna!Tulokset.Clear
    'Tyhjennetään lista.

    'Käydään tulostaulu läpi rivi kerrallaan.
    Do Until MunHaku.EOF
        Rivi = ""
        'Tulostaulun kenttien numerointi alkaa nollasta.
        'Käydään tulostaulun yhden rivin kentät läpi.
        For i = 0 To MunHaku.Fields.Count - 1
            Rivi = Rivi & MunHaku.Fields(i) & " "
        Next
        'Lisätään rivi listaan.
        tulosikkuna!Tulokset.AddItem Rivi
        'Siirrytään osajoukon seuraavalle riville.
        MunHaku.MoveNext
    Loop
    'Suljetaan osajoukko.
    MunHaku.Close
    'Suljetaan tietokanta.
    MunKanta.Close
    tulosikkuna.Show
Exit Sub

virhe:
    MsgBox "Hakuavain väärin.", 48, "Virhe"
    Exit Sub
End Sub
```

Kun ohjelmasta tehdään EXE-versio, on kirjoitettava INI-tiedosto, jossa kerrotaan, mistä löytyvät tarvittavat tiedot käyttää muuta kuin Microsoft Access -tietokantaa. INI-tiedoston polku asetetaan esim. Form_Loadissa **SetDataAccessOption**-lauseella. Asetusta ei tarvitse tehdä, jos INI-tiedosto on saman niminen kuin ohjelma.

```
Sub Form_Load ()
  SetDataAccessOption 1, ".\kir.ini"
End Sub
```

Jos Paradox-taulukoissa halutaan käyttää muuta kuin ASCII-järjestystä, pitää VB.INI-tiedoston lopussa olla sama teksti kuin KIR.INI:n lopussa eli tieto lajittelukielestä.

Esimerkkiohjelman KIR.INI on seuraavanlainen:

```
[Installable ISAMs]
Paradox 3.X=D:\WINDOWS\SYSTEM\pdx110.DLL

[Paradox ISAM]
;mahdolliset muut määritykset
```

```
CollatingSequence=Swedish-Finnish      ;Tässä se järjestys, jolla taulukot
                                       ;on tehty.
```

Kirjallisuutta

Language Reference Microsoft Corporation, 1993

Professional Features Book 1 & 2 Microsoft Corporation, 1993

Programmer's Guide Microsoft Corporation, 1993

Dittrich, Stefan *Visual Basic*; Tecnopress, 1992

Majaniemi, Antti *Visual Basic : alkeita*; Kymdata, 1993

MikroPC-lehti 10-12/1993, 1-4/1994

Pölhö Jukka *Visual Basic -käsikirja*; Pagina, 1992

Pölhö Jukka *Visual Basic -ohjelmointi*; Pagina, 1993

Tietokone-lehti 2-4, 6-7, 8/1994

Young, Michael J. *Visual Basic - game programming for Windows*; Microsoft Press, 1992

Liite 1. Ohjaimet ja niiden yleisimpiä ominaisuuksia

Valitaan haluttu ohjain aktiiviseksi. (Pointer)

KUVAKEHYS (Picture Box Control)

Align = 1 Kuvakehyksen leveys seuraa automaattisesti lomakkeen leveyttä ja kehys pysyy lomakkeen yläreunassa.

Align = 2 Tapahtuu sama kuin edellä, mutta lomakkeen alareunassa.

OTSIKKO (Label Control)

Caption	Otsikon teksti sijoitetaan tähän.
AutoSize = True	Otsikon koko mukautuu Caption-tekstin mukaan.
WordWrap = True	Otsikon koko mukautuu pystysuunnassa, muuten vaakasuunnassa.
Alignment = 0	Teksti vasempaan laitaan.
Alignment = 1	Teksti oikeaan laitaan.
Alignment = 2	Teksti keskelle.

TEKSTILAATIKKO (Text Box Control)

Text	Teksti, jota käyttäjä voi muuttaa.
Multiline = False	Tekstilaatikossa vain yksi rivi.
Multiline = True	Tekstilaatikossa useampi rivi.

KEHYS (Frame Control)

Visible = 0 (False)	Kaikki kehykseen sijoitetut ohjaimet ovat näkymättömiä.

NAPPI (Command Button Control)

Enabled = False	Painike ei ole käytössä.
Cancel = True	Nappi on lomakkeen peruutusnappi.
Default = True	Vain yksi nappi voi olla lomakkeella oletusnappi.

VALINTARUUTU (Check Box Control)

Value = 0	Ruutua ei ole rastittu.
Value = 1	Ruutu on rastittu.
Value = 2	Ruutu on harmaa (mitätöity).

VALINTANAPPI (Option Button Control)

Value = False	Nappia ei ole valittu.
Value = True	Nappi on valittu.

Vain yksi valintanappi lomakkeella tai kehyksessä voi olla valittuna.

YHDISTELMÄLISTA (Combo Box Control)

Listindex	**I**lmoittaa valitun luettelokohdan.
Listindex = -1	Mitään ei ole valittu.

Text Sisältää valitun kohdan tekstin. Mahdollinen oletusarvo korvautuu ajon aikana valitulla tekstillä.

LISTA (List Box Control)

VIERITYSPALKKI (Horizontal Scroll Bar Control)

VIERITYSPALKKI (Vertical Scroll Bar Control)

Max, Min	Vierityspalkin maksimi- ja minimiarvot.
Smallchange	Kuinka paljon liikutaan nuolesta painamalla.
Largechange	Kuinka paljon liikutaan palkkia painamalla.

AJASTIN (Timer Control)

Interval	Kuinka usein ajastin reagoi. Arvo millisekunteja.

LEVYASEMALISTA (Drive List Box Control)

HAKEMISTOLISTA (Directory List Box Control)

TIEDOSTOLISTA (File List Box Control)

KUVIO (Shape Control)

Shape = 0	suorakulmio
Shape = 1	neliö
Shape = 2	ellipsi
Shape = 3	ympyrä
Shape = 4	pyöreäreunainen suorakulmio
Shape = 5	pyöreäreunainen neliö

VIIVA (Line Control)

KUVA (Image Control)

Stretch = False	Ohjain mukautuu bittikartan kokoon.
Stretch = True	Bittikartta mukautuu ohjaimen kokoon.

TAULUKKO (Grid Control)

Cols, Rows	Pysty- ja vaakasarakkeiden määrä.
FixedCols, FixedRows	Kiinteiden sarakkeiden ja rivien määrä.
GridLines	Ovatko taulukon solujen rajaviivat näkyvissä vai eivät.
HighLight = True	Valitut solut ovat korostettuja.
LeftCol, TopRow	Määrittävät ja palauttavat vasemmanpuoleisimman ja ylimmän ei-kiinteän sarakkeen ja rivin.

ColAlignment[*lomake*!]*taulukko.* **ColAlignment**(*indeksi*)[*numero*]
Ominaisuus määrää tekstin asettumisen *indeksin* määrittämässä sarakkeessa.

Numeron arvot:

0	teksti vasemmalle
1	teksti oikealle
2	teksti keskelle
3	käytetään sarakkeen **ColAlignment** -arvoa.

FixedAlignment	Määrää tekstin asettumisen sarakkeen kiinteillä soluilla. Syntaksi ks. edellä.
Col, Row	Palauttavat ja asettavat aktiiviset sarakkeet ja rivit.

OLE (OLE Client Control)

Action = 0	Luo upotetun olion.
Action = 1	Luo linkitetyn olion tiedostosta.
Action = 4	Kopioi olion leikepöydälle
Action = 5	Kopioi leikepöydältä olioon.
Action = 6	Päivittää olion.
Action = 7	Avaa olion operaatiolle, esim. editoitavaksi.
Action = 9	Sulkee yhteyden.
Action = 10	Tuhoaa olion ja vapauttaa muistin.
Action = 11	Tallettaa olion.
Action = 12	Lataa talletetun olion tiedostosta.
Action = 14	Näyttää Insert Object -dialogin.
Action = 15	Näyttää Paste Special -dialogin.
Class	Määrittelee upotetun olion tyypin, esim. "ExelWorksheet".
DisplayType	Näytetäänkö olion sisältö vai ikoni.
SourceDoc	Arvo on linkitettävän tiedoston nimi polkuineen.
SourceItem	Arvo määrää, mikä osa tiedostosta linkitetään.
UpdateOptions = 0	**Automaattinen** eli olio päivitetään aina kun tieto muuttuu.
UpdateOptions = 1	**Jäädytetty** eli olio päivittyy aina, kun käyttäjä tallettaa linkitetyn tiedoston linkitetyssä ohjelmassa.
UpdateOptions = 2	**Manuaalinen** eli olio päivitetään vain, kun asetetaan **Action = 6**.
Verb	Määrää operaation aktiiviselle oliolle.
Verb = 0	Olion oletusoperaatio.
Verb = -1	Aktivoi olion editoitavaksi. Mahdollisuus in-place -editointiin, jos upotettu ohjelma tukee sitä.
Verb = -2	Avaa upotetun ohjelman omassa ikkunassaan.
Verb = -3	Piilottaa olion luoneen ohjelman.

ANIMAATIO (Animated Button Control)

Tähän ohjaimeen voidaan liittää sarja kuvia.

Frame	Määritellään ruudun (kuvan) järjestys animaatiossa.

VAKIODIALOGI (Common Dialog Control)

Käytettävissä ovat vakiodialogit tiedostojen avaamista ja tallentamista varten, värien ja fonttien valintaa varten sekä tulostamista varten.

MITTARI (Gauge Control)

PYLVÄIKKÖ (Graph Control)

NÄPPÄINTILA (Key Status Control)

Style = 0	Caps Lock
Style = 1	Num Lock
Style = 2	Insert
Style = 3	Scroll Lock
Value	Asettaa ja palauttaa ohjaimen ja **Style**-arvon määrämän näppäimen tilan.

TIETOLIIKENNE (Communications Control)

MASKITEKSTIRUUTU (Masked Edit Control)

Mask	Syöttötiedot suodattava maski. Esim "##.##.##"
#	numero
&	mikä tahansa merkki
A	aakkosnumeerinen merkki
?	aakkosmerkki

KUVALEIKKURI (Picture Clip Control)

Clip	Palauttaa leikatun kuva-alueen.

HYRRÄ (Spin Button Control)

Delay	Kuinka monen millisekunnin välein hyrrä reagoi.

KOLMIULOTTEINEN VALINTARUUTU (3D Check Box Control)

KOLMIULOTTEINEN KEHYS (3D Frame Control)

KOLMIULOTTEINEN VALINTANAPPI (3D Option Button Control)

KOLMIULOTTEINEN NAPPI (3D Command Button Control)

KOLMIULOTTEINEN PANEELI (3D Panel Control)

PAINIKERYHMÄ (3D Group Push Button Control)

TIETOKANTA (Data Control)

(CrystalReport control)

HIERARKINEN LISTA (Outline Control)

Liite 2. Erikoisoliot

SOVELLUS (App Object)

EXEName	Ohjelmointiympäristössä arvo on projektin nimi. Kun projekti käännetään EXE-tiedostoksi, arvoksi tulee valittu nimi ilman tarkennetta.
Title	Sovelluksen nimi Task Managerissa.
Path	Sovelluksen polku.
PrevInstance	Arvo EXE-tiedostolla True, jos saman sovelluksen esiintymä on jo ajossa, muuten False.

LEIKEPÖYTÄ (Clipboard Object)

SetText Laittaa tekstin leikepöydälle.

Clipboard.SetText *data* [,*format*]

data Teksti, joka laitetaan leikepöydälle.

format Jokin seuraavista:

Vakio	Arvo	Muoto
CF_LINK	&HBF00	DDE keskusteluinformaatio
CF_TEXT	1	teksti

Jos muotoa ei ole annettu, muoto on CF_TEXT.

GetText Liittää tekstin leikepöydältä.

Clipboard.GetText ([*format*])

SetData Laittaa kuvan leikepöydälle.

Clipboard.SetData *data*,[*format*]

data Kuva, joka laitetaan leikepöydälle.

format Jokin seuraavista:

Vakio	Arvo	Muoto
CF_BITMAP	2	Bitmap (.BMP)
CF_METAFILE	3	Metafile (.WMF)
CF_DIB	8	Device-independent bitmap
CF_PALETTE	9	Color palette

Jos *format* on 0 tai sitä ei ole annettu, SetData metodi määrää sen automaattisesti.

GetData Liittää kuvan leikepöydältä.

Clipboard.GetData ([*format*])

NÄYTTÖ (Screen Object)

Näyttö-oliota käytetään lomakkeiden paikan määräämiseen ruudulla sekä ajon aikana kursorin kontrolloimiseen ohjelman lomakkeiden ulkopuolella. Esimerkiksi **MousePointer**-ominaisuuden avulla voidaan kursori haluttaessa muuttaa tiimalasiksi. **Screen.ActiveObject** kertoo, mikä ohjain lomakkeella on aktiivinen. **Screen.ActiveForm** kertoo, mikä lomake (ikkuna) on aktiivinen. **Font** ja **FontCount** kertovat, mitkä fontit ovat näytölle käytössä.

TULOSTIN (Printer Object)

Tulostin-oliota käytetään kontrolloimaan tulostettavaa tekstiä ja grafiikkaa ja lähettämään tulostus suoraan oletuskirjoittimelle.

VIRHEENETSINTÄ (Debug Object)

Virheenetsintä-olio kontrolloi virheenetsintäikkunaa ajon aikana.

Liite 3. Lisää ominaisuuksia

DrawWidth

Tämä ominaisuus on lomakkeella, kuvakehyksellä ja Printer-oliolla. Se määrää viivan paksuuden käytettäessä graafisia metodeja (esim. Line, Circle). Arvo voi olla 1 - 32 767 pikseliä. Oletusarvo on 1.

DrawMode

Määrää graafisilla metodeilla tuotettujen kuvioiden ulkoasun ja viiva- ja kuvio-ohjaimen ulkoasun. Ominaisuuden arvot ovat 1-16.

DrawStyle

Tämä ominaisuus on lomakkeella, kuvakehyksellä ja Printer-oliolla. Se määrää viivan laadun käytettäessä graafisia metodeja (esim. Line, Circle). Ominaisuuden arvot ovat 0-6.

Tag

Tätä ominaisuutta voidaan käyttää tiedon tallentamiseen (merkkijono) eikä se vaikuta ohjaimen muihin toimintoihin.

Enabled

Jos arvo on nolla, ohjain ei reagoi hiiri- ja näppäintapahtumiin.

Tabstop

Jos arvo on nolla, ohjain ohitetaan sarkainnäppäimellä siirryttäessä.

Tabindex

Arvo määrää järjestyksen, jossa sarkainnäppäimellä siirrytään ohjaimesta toiseen. Alunperin järjestys on ohjainten sijoitusjärjestys lomakkeella.

Index

Kaikilla ohjaintaulukoilla on tämä ominaisuus. Jos kopioi esim. nappeja, Visual Basic kysyy, halutaanko niistä tehdä taulukko. Jos halutaan taulukko, on siihen sisältyvillä ohjaimilla oltava sama nimi (Name). Index on ohjaintaulukon alkion järjestysnumero.

Dragmode

Jos arvona on Automatic - 1, ohjainta voi automaattisesti vetää hiirellä. Vedettyä ohjainta ei kuitenkaan voi pudottaa ilman eri koodia.

WindowState

Määrää lomakkeen (ikkunan) tilan ajon aikana.

[*form*!]**WindowState**[= *state*]

state	
0	Normaali (oletus)
1	Minimoitu
2	Maksimoitu

Liite 4. Tapahtumia

DragDrop

Tapahtuma syntyy kun drag-and-drop -operaatio on suoritettu ja jokin ohjain on viety toisen ohjaimen tai lomakkeen päälle ja hiiren nappi on vapautettu.

Sub *ctlname*_**DragDrop (**[*Index* **As Integer**,] *Source* **As Control**, *X* **As Single,** *Y* **As Single)**

Index	Ohjaimen tunniste, jos kyseessä ohjaintaulukko.
Source	Ohjain, jota vedetään.
X, Y	Hiiren vaaka- (X) ja pystysijainti (Y) kohdelomakkeessa tai -ohjaimessa.

Tätä tapahtumaa käytetään esim. jos halutaan vaikka siirtää vedetty ohjain uuteen paikkaan tai kopioida tiedosto paikasta toiseen.

DragOver

Tapahtuma syntyy kun drag-and-drop -operaatio on käynnissä. Tällä tapahtumalla voi tutkia, milloin hiiri saapuu kohteeseen, lähtee kohteen päältä tai on juuri kohteen päällä.

Sub *ctlname*_**DragOver (**[*Index* **As Integer**,] *Source* **As Control**, *X* **As Single**, *Y* **As Single**, *State* **As Integer)**

Index	Ohjaimen tunniste, jos kyseessä ohjaintaulukko.
Source	Ohjain, jota vedetään.
X, Y	Hiiren vaaka- (X) ja pystysijainti (Y) kohdelomakkeessa tai -ohjaimessa.
State	Vedettävän ohjaimen suhde kohteeseen:
0 (Enter)	Ohjainta vedetään kohteeseen.
1 (Leave)	Ohjainta vedetään kohteesta.
2 (Over)	Ohjain on siirretty kohteen päälle.

GotFocus

Tapahtuma syntyy, kun ohjain aktivoituu (saa fokuksen).

Sub *ctlname*_**GotFocus** ([*Index* **As Integer**])

LostFocus

Tapahtuma syntyy, kun ohjain menettää fokuksen.

Sub *ctlname*_**LostFocus (**[*Index* **As Integer**]**)**

MouseDown, MouseUp

Tapahtuma syntyy kun käyttäjä painaa hiiren nappia (MouseDown) tai vapauttaa sen (MouseUp).

Sub *ctlname*_**MouseDown (**[*Index* **As Integer**,] *Button* **As Integer**, *Shift* **As Integer**, *X* **As Single**, *Y* **As Single)**

Button	Button on se hiiren nappi, jota on painettu tai joka on vapautettu.
Shift	Onko painettu myös nappeja Shift, Ctrl tai Alt.
X, Y	Hiirikursorin paikka.

Click, DblClick

Sub *ctlname*_**Click** (*Index* **As Integer**)

KeyDown, KeyUp

Tapahtuma syntyy, kun käyttäjä painaa (KeyDown) tai vapauttaa (KeyUp) näppäimen, kun ohjaimella on fokus.

Sub *ctlname*_**KeyDown (**[*Index* **As Integer**,]*KeyCode* **As Integer**, *Shift* **As Integer)**

KeyCode Näppäinkoodi esim.KEY_F1 (F1 key) ja KEY_HOME (Home key). Vakiot määritelty tiedostossa CONSTANT.TXT.

KeyPress

Tapahtuma syntyy, kun käyttäjä painaa ANSI-näppäintä.

Sub *control*_**KeyPress (**[*Index* **As Integer**,]*KeyAscii* **As Integer)**

KeyAscii Palauttaa ANSI-näppäinkoodin (a standard numeric ANSI keycode).

Change

Aina, kun esimerkisi tekstiruudun teksti muuttuu.

Sub *ctlname*_**Change (***Index* **As Integer)**

Load

Aina, kun lomake ladataan muistiin.

Sub Form_Load ( )

Unload

Aina, kun lomake on poistumassa muistista

Sub Form_Unload (*Cancel* **As Integer)**

Cancel Kun arvoksi asetetaan jokin nollasta poikkeava arvo, lomake ei poistu muistista.

Liite 5. Metodeja

Paitsi ominaisuuksia ja tapahtumia ohjaimiin kuuluu myös metodeja. Metodin yleinen syntaksi on: [*Lomake*!][*Ohjain.*]**metodi**[*argumentit*] Jokaisella ohjaintyypillä on etukäteen määrätyt metodit. Metodi voi myös palauttaa arvon. Metodeja ei voi tehdä itse.

ZOrder

Tuo ohjaimen edustalle tai asettaa sen taustalle.

[*object.***]ZOrder [***position***]**

position = 0 tai sitä ei anneta, ohjain asetetaan edustalle.
position = 1 Ohjain asetetaan taustalle.

Move

[object.**]Move** left**[**, top**[**, width**[**, height**]]]**

Esim. teksti.**Move** teksti.Left + x, teksti.Top + y

object	Lomake tai ohjain, jota siirretään.
left	Arvo osoittaa objektin vasemman kulman vaakakoordinaatin.
top	Arvo osoittaa objektin vasemman kulman pystykoordinaatin.
width	Arvo osoittaa objektin uuden leveyden.
height	Arvo osoittaa objektin uuden korkeuden.

Vain argumentti *left* on pakollinen. Kuitenkin, jos haluaa määritellä jonkin muun argumentin, tulee määritellä kaikki sitä ennen tulevat argumentit.

Scale

Määritellään olion koordinaatisto.

[*object.***]Scale [**(*x1, y1*) **-** (*x2, y2*)**]**

object	Lomake (ei MDI), kuvakehys tai Printer-olio, jonka koordinaatistoa käsitellään.
x1, y1	Olion vasen yläkulma.
x2, y2	Olion oikea alakulma.

Line

[*object.*]**Line** [[**Step**](*x1, y1*)] - [**Step**](*x2, y2*) [,[*color*][,**B**[**F**]]]

object	Lomake (ei MDI), kuvakehys tai Printer-olio, johon piirretään.
Step	Avainsana, joka ilmaisee, että kohta on suhteessa edelliseen piirrettyyn kohtaan.
x1, y1	Viivan alkupiste.
Step	Avainsana, joka ilmaisee, että loppupiste on suhteessa alkupisteeseen.

```
Line (100, 200)-(150, 250)
Line (100, 200)-Step(50, 50) 'Sama kuin ylläoleva.
```

x2, y2	Viivan loppupiste.
color	Viivan RGB-värinumero.
B	Suorakulmiota piirrettäessä tarvitsee antaa vain vasen yläkulma ja oikea alakulma.
F	Tarkoittaa, että suorakulmio väritetään samalla värillä kuin viiva.

Circle

[*object.*]**Circle** [**Step**](*x, y*), *radius*[,[*color*] [,[*start*] [,[*end*] [,*aspect*]]]]

object	Lomake (ei MDI), kuvakehys tai Printer-olio, johon piirretään.
Step	Avainsana, joka ilmaisee, että keskipiste on suhteessa *object*:in koordinaatteihin (CurrentX -, CurrentY -ominaisuudet).
x, y	Keskipisteen koordinaatit.
radius	Ympyrän, ellipsin tai kaaren radiaani.
color	Ympyrän, ellipsin tai kaaren RGB-värinumero.
start, end	Kun piirretään osaympyrä tai -ellipsi, start ja end ilmoittavat kaaren alun ja lopun radiaaneina.
aspect	Litteys; mitä isompi luku sitä litteämpi.

Point

Palauttaa RGB-värin halutusta lomakkeen tai kuvakehyksen pisteestä.

[*object.***]Point(***x,y***)**

object	Lomake (ei MDI) tai kuvakehys, jossa piste on.
x, y	Pisteen koordinaatit.

PSet

Asettaa tietyn pisteen halutun väriseksi.

[*object.***]PSet [Step]**(*x, y*)**[,***color***]**

object	Lomake (ei MDI), kuvakehys tai Printer-olio, johon piste piirretään.
Step	Avainsana, joka ilmaisee, että kohta on suhteessa *object*:in koordinaatteihin (CurrentX -, CurrentY -ominaisuudet).
x, y	Pisteen koordinaatit.
color	Pisteen RGB-värinumero.

Cls

Poistaa lomakkeelle tai kuvakehykseen ajon aikana kirjoitetun tekstin (Print) ja piirretyn grafiikan (esim. Line, Circle).

[*object.***]Cls**

object	Lomake tai kuvakehys, johon toimenpide kohdistetaan

Print

Kirjoittaa tekstiä esim. lomakkeelle tai kuvakehykseen.

[*object.***]Print [{Spc(***n***) | Tab(***n***)}][***expressionlist***][{; | ,}]**

object	Lomake (ei MDI), kuvakehys, Printer-, tai Debug-olio, johon teksti kirjoitetaan.
Spc(*n***)**	Funktio, joka lisää n välilyöntiä.
Tab(*n***)**	Siirtyy n:nteen sarakkeeseen ennen tekstin kirjoittamista.
expressionlist	Kirjoitettava teksti. Voi olla useita esim. välilyönnillä erotettuina.
{; | ,}	Määrää kursorin paikan. Jos ei kumpaakaan merkkiä, seuraava teksti alkaa uudelta riviltä. Puolipiste sijoittaa kursorin heti edellisen merkin jäl-

keen ja pilkku siirtää kursorin seuraavaan kirjoitusalueeseen.

```
Print "koe"; "koe";
Print Chr(10)
Print "koe", "koe",
Print Chr(10)
Print "koe"; "koe"
```

```
Tulostus:

koekoe
koe   koe
koekoe
```

Liite 6. Ohjainten oletusominaisuudet

CheckBox	Value
ComboBox	Text
CommandButton	Value
DirectoryListBox	Path
DriveListBox	Drive
FileListBox	FileName
Frame	Caption
Grid	Text
Horizontal ScrollBar	Value
Image	Picture
Label	Caption
Line	Visible
ListBox	Text
Menu	Enabled
OLE client	Action
OptionButton	Value
Picture	Picture
Shape	Shape
TextBox	Text
Timer	Enabled
Vertical ScrollBar	Value

Liite 7. Ongelman ratkaisu 1. esimerkkiin

Kananmunien paistaminen

Laitetaan kananmunat pannulle. Katsotaan kelloa. Se on 12.10.

Otetaan kananmunat pois pannulta. Katsotaan kelloa. Se on 12.13.

Kananmunat ovat paistuneet vasta 12.13 - 12.10 = 3 minuuttia.

Laitetaan kananmunat takaisin pannulle. Katsotaan taas kelloa. Se on 12.20.

Otetaan kananmunat pois pannulta kello 12.25.

Yhteensä munat ovat paistuneet 3 + (12.25 - 12.20) = 3 + 5 = 8 minuuttia.

Sekuntikellon toiminta

Kun kello pannaan käyntiin kysytään, paljonko kello on?

Vastaus: 20

Kuluu aikaa.

Paljonko kello nyt on?

Vastaus: 36

Paljonko on kulunut aikaa? Mihin viisaria siirretään?

Vastaus: 0 + (36 - 20) = 16 sekuntia, viisari siirretään kohtaan 16.

Pysäytetään kello. Otetaan talteen senhetkinen aika, joka on esim. 0 + (38 - 20) = 18.

Laitetaan kello uudelleen käyntiin. Paljonko kello on?

Vastaus: 56

Kuluu aikaa.

Paljonko kello nyt on?

Vastaus: 57

Mihin viisaria siirretään?

Vastaus: 18 + (57 - 56) = 18 + 1 = 19, viisari siirretään kohtaan 19.

Hakemisto

!

!, 45

#

#, 16

&

&, 35
&-operaattori, 15

'

', 9

+

+-operaattori, 15

:

:, 9

<

<, 35
<=, 35
<>, 35

=

=, 35

>

>, 35
>=, 35

?

?-, 43

\

\, 35

A

access, 65
Action, 52; 53; 54; 55
ActiveForm, 59
Add Watch, 43
AddItem, 20; 22; 51
Ajastin, 27
Alias, 60
Align, 24
aloituslomake, 13
And, 35
Append, 66
APULAINE.MAK, 23
Asc(merkkijono), 13
AutoRedraw, 32
AutoSize, 19

B

BAS, 5
Binary, 67
Binääritiedostot, 67
Breakpoint, 43
Button, 29
ByVal, 40; 41; 61

C

Call, 40
Cancel, 32; 45
CancelError, 53
CASCADE, 60
CF_BITMAP, 69
CF_BOTH, 54
CF_DIB, 69
CF_LINK, 68
CF_METAFILE, 69
CF_PALETTE, 69
CF_PRINTERFONTS, 54
CF_SCREENFONTS, 54
CF_TEXT, 68
ChDir, 23
ChDrive, 23
CheckBox, 25
Chr$(ANSI-numero), 13
Clear, 21
Close, 65
CMDIALOG.VBX, 52
Color, 54
Color-ominaisuus, 54
Cols, 51
Columns, 21
ComboBox, 20
Connect, 73
Const, 17
CONSTANT.TXT, 17; 60
Control, 3
CONTROL.MAK, 19; 26
Controls, 46
Copies, 55
Copy, 67
Count, 46
CurDir$, 24
Currency, 15
Cut, 67

D

D_AND_D.MAK, 30
D_AND_D2.MAK, 30
DataBaseName, 73
DataField, 73
DataManager, 73
DataSource, 73
DDE, 69
Debug-ikkuna, 43
declarations, 16
Declare, 60; 61; 63
Default, 32; 45
dialogi, 45
DialogTitle, 52
Dim, 13; 36
DirectoryListBox, 22
DLL, 60
DLLINI.MAK, 61
Do Until, 36
Do While, 36
Do...Loop, 36
Drag, 30
Drag And Drop, 29
DragIcon, 30
DragMode, 29
DragOver, 30
DriveListBox, 22
Dropdown Combo, 20
Dropdown List, 20
Dynaamiset taulukot, 38
Dynaset, 74

E

Empty, 16
Empty Value, 41
Enabled, 27
erikoisoliota, 4
erityinen, 49
Err, 57
Error, 57
esiintymä, 46
Events, 3
EXE-versio, 10
Exit Do, 36
Exit For, 36
Exit Function, 41
Exit Sub, 40

F

file, 65
FileLen, 66
FileListBox, 22
FileName, 53; 54
filenumber, 65
Filter, 52
FilterIndex, 53
FixedCols, 51
FixedRows, 51
Flags, 54
Font, 54
Fontinvaihtodialogi, 54
For...Next, 36
Form_Load(), 13
Form_Unload(Cancel As Integer), 13
Forms, 46
FreeFile, 66
FRM, 5
FromPage, 55
Function, 40

G

general-osa, 9
Generic, 49
Get, 65; 67
GetData(), 68
GetFormat, 69
GetPrivateProfileString, 61
GetText(), 68
Global, 13
Grid, 51

H

hakemisto-, 22
hDc, 55
HELP.MAK, 58
HelpContextID, 58
HERATYS.MAK, 27
Hide, 13; 45
HideSelection, 20

I

IDOK, 56
If, 35
If...TypeOf, 50
Image, 24
INI-tiedosto, 61; 75
InitDir, 53
Input, 66
Input$, 66
InputBox, 56
InputBox$, 56
Installable ISAMs, 75
Interval, 27
Is, 49
ISAM, 75
IsDate, 15
IsEmpty, 15
IsNull, 15
IsNumeric, 15

K

keskitys-ominaisuus, 24
KeyAscii, 31
KeyDown, 31
KeyPress, 31; 32
KeyPreview, 31
KeyUp, 31
KIR.INI, 75
KIR.MAK, 73
kommenttimerkki, 9
koodi-ikkuna, 8; 10
Kopioi, 67
Kuva, 24
Kuvakehys, 24
KUVAT.MAK, 25

L

Label, 19
Largechange, 26
LBound, 39
LCase$(merkkijono), 14
Left$(jonolauseke, n), 14
LEIKE.MAK, 67
Leikepöytä, 67
Leikkaa, 67
Len(mekkijono), 14
Levyasema-, 22
Lib, 60
Liitä, 68
Line, 29
LinkItem, 70
LinkMode, 70
LinkRequest, 70
LinkSend, 71
LinkTopic, 69
LISAAMIN.MAK, 37
Lista, 21
LISTA.MAK, 21
ListBox, 21
ListCount, 21; 22
ListIndex, 21
Load, 37; 45
LoadPicture, 25
lock, 65
LOF, 66
lomakkeesta, 4
Lomakkeet, 13
Loop While Not, 36

M

MAK-tiedosto, 5
Matriisit, 38
max, 26
MDI, 58
MDIChild, 58
Me, 47; 59
Menu, 47
Menu Design, 47
MessageBox, 55
Methods, 3
metodeja, 3
Mid$(jonolauseke, i, n), 14
min, 26
MkDir, 24
Mod, 35
modal, 45
mode, 65
Modeless, 45
MouseDown, 29
MouseMove, 29
MouseUp, 29
MsgBox-funktio, 55
MsgBox-lauseke, 55
MultiLine, 20
MultiSelect, 21
muuttujatyypit, 13

N

Name, 6
NAPPAIN.MAK, 31
New, 46
Next, 36
Not, 35; 50
Nothing, 47
Now, 16
NUL, 61
Null, 16

O

ohjaimiksi, 3
Ohjaintaulukko, 37
olio, 4
Olio-osoittimet, 49
ominaisuuksia, 4
ominaisuuksille, 3
ominaisuusikkuna, 28
On Error, 57
Open, 52; 65
OpenDatabase, 74
OptionButton, 25
Or, 35
Otsikko, 19
Output, 66

P

PAADLL.C, 62
Paint, 32
PAIVA.MAK, 16
palsta, 21
Paradox, 73
Paste, 68
Paste Link, 69; 71
Pattern, 23
PD_PAGENUMS, 55
PD_RETURNDC, 55
Peräkkäistiedostot, 66
Picture, 51
PictureBox, 24
Preserve, 37; 39
Print, 55; 66
private, 39
Properties, 3
PSet, 29
public, 39
Put, 65; 67
PVMDLL.C, 61
PVMDLL.DEF, 62
PVMLIB.H, 62

Q

QueryUnload, 59

R

Random, 65
reclen, 65
RecordSource, 73
ReDim, 37; 38
RemoveItem, 21; 51
Resume, 57
Resume Next, 57
Right$(jonolauseke, n), 14
RmDir, 24
Rows, 51
rtf, 58
RUUDUKKO.MAK, 52

S

Save As, 53
ScrollBar, 26
ScrollBars, 20
Seek (tiedostonro), 66

Seek tiedostonro, kohta, 66
SEK.MAK, 5
Select Case, 35; 50
Selected, 22
SelEndCol, 51
SelEndRow, 51
SelLength, 20
SelStart, 20
SelStartCol, 51
SelStartRow, 51
SelText, 20; 67
Sequential, 66
Set, 49; 50
SetData, 67
SetDataAccessOption, 75
SetFocus, 45
SetText, 67
Shift, 29
Show, 13; 45
Simple Combo, 20
Smallchange, 26
Sorted, 20
Specific, 49
SQL, 73
Static, 13
Step, 36
Str$(luku), 14
Stretch, 24
String, 13
Sub, 40
Sub Main, 13
Suorasaantitiedostot, 65
Syöttödialogi, 56

T

TabIndex, 45
TabStop, 32
Tag, 59
talletusdialogi, 53
tapahtuma, 4
tapahtumia, 3
Taulukko-ohjain, 51
Tekstilaatikoita, 20
Text, 51
TextBox, 20
tiedostolista, 22
Tiedostolistat, 22
Tiedoston avaus, 52
Tietokanta, 73
Tietueet, 16
TILE_HORIZONTAL, 60
Time, 16
Timer, 27
To, 36; 38
ToPage, 55
Tulostusdialogi, 55
Type, 16

U

UBound, 39
UCase$(merkkijono), 14
Unload, 37; 45

V

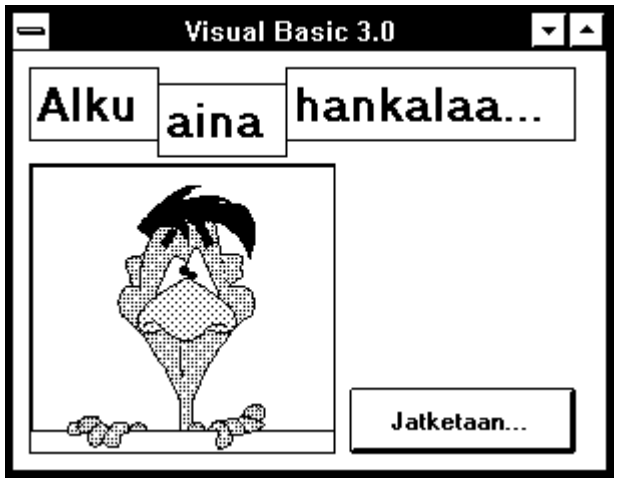
Vakiodialogit, 52
Vakiokokoinen merkkijono, 13
vakiot, 17
Val(merkkijono), 13
Valikot, 47
valintanapit, 26
Valintaruutu, 25
Value, 25; 26
Variant, 15; 41
VB.INI, 75
VBX-tiedosto, 5; 11
venytys-ominaisuus, 24
Vierityspalkit, 26
Viestidialogi, 55
Visible, 37
Värinvaihtodialogi, 54

W

WordWrap, 19
WritePrivateProfileString, 61

Y

Yhdistelmälista, 20
Yksiulotteiset taulukot, 36
yleinen, 49

Kaarina Mäenpää

Visual Basic 3.0