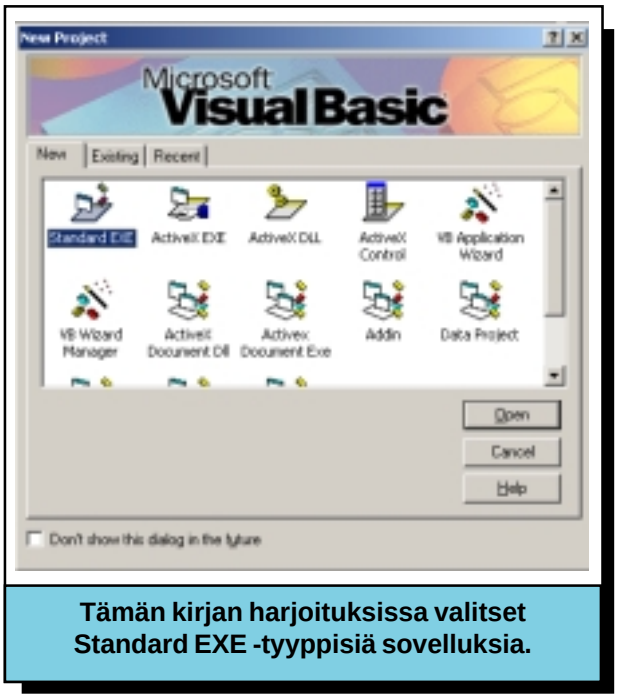
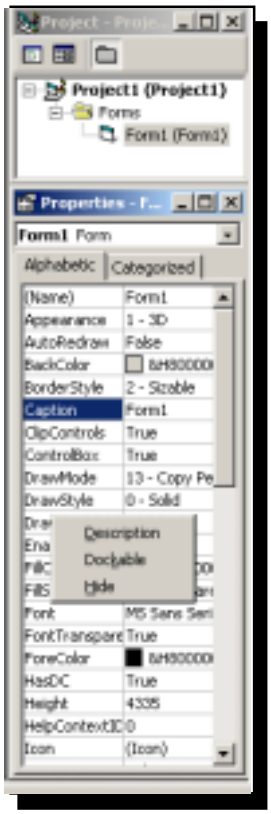
Sisällysluettelo

1. OHJELMOINTIYMPÄRISTÖ

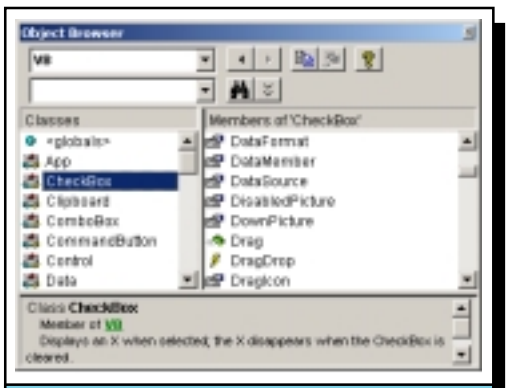
Visual Basic -sovellus koostuu lomakkeista ja moduuleista, jotka yhdessä muodostavat projektin. Käynnistettäessä VB ensimmäistä kertaa muodostat automaattisesti uuden projektin valitsemalla Standard EXE. Saman valintaikkunan saat valinnalla File | New Project. Käytä jatkossa toista edellä kuvatuista tavoista aloittaessasi uuden sovelluksen.



Tämän kirjan harjoituksissa valitset Standard EXE -tyyppisiä sovelluksia.



VB-ympäristön kaikkia ikkunoita voi siirtää ja niiden kokoa muuttaa. Projektinhallinta- (Project Explorer) ja Ominaisuudet ikkunan (Properties Window) lisäksi kannattaa Objektiselain (Object Browser) tuoda näkyviin. Objektiselaimella voit katsella sovelluksesi ominaisuuksien lisäksi tarvittaessa kaikkia VB:n tuntemien objektien ominaisuuksia, tapahtumia ja metodeja.



Huomaa, kuinka ominaisuudet, tapahtumat ja metodit näkyvät erilaisina kuvakkeina.

Seuraavassa muutamia hyödyllisiä koodaamista ja testaamista nopeuttavia näppäilyjä, jotka kannattaa ottaa heti alussa käyttöön.

CTRL-N	***Uusi projekti***
CTRL-O	***Avaa vanha projekti***
F2	***Objektiselain***
F5	***Ohjelman suorittaminen***
F7	***Lomake aktiiviseksi***
SHIFT-F7	***Koodi aktiiviseksi***
CTRL-BREAK	***Ohjelman keskeytys***
ALT-F4	***Sovelluksen lopetus***

Ohjelmointiympäristön käytöstä ja näppäilyistä lisää seuraavan luvun esimerkkiohjelmassa.

2. TIETOTYYPIT JA NIMEÄMINEN

Tietokoneohjelmissa kaikki prosessoitava tieto löytyy tietokoneen keskusmuistista. Ohjelmoija käyttää kuitenkin harvemmin keskusmuistipaikan järjestysnumeroa ohjelmissaan, vaan itse tarkoituksenmukaisesti nimeämiään muuttujia. Muuttuja on tiedon tallennuspaikka ja sen sisältö on yleensä numeerista tai aakkosnumeerista tietoa. Jos muuttujan arvoa muutetaan, aiempi tieto katoaa lopullisesti.

Visual Basicissa muuttu jat esitellään joko koko ohjelman suorituksen ajan muistia varaavina globaaleina muuttujina ohjelman alussa (tai moduuleissa) tai aliohjelmissa lokaaleina muuttujina varatun sanan Dim jälkeen. Yleensä lokaaleja muuttujia kannattaa suosia, koska ne varaavat keskusmuistia ainoastaan aliohjelman suorituksen ajan. Kun aliohjelman suoritus loppuu, muuttujan varaama keskusmuistitila vapautuu ja muuttujan tieto häviää. Jos tietoa halutaan käyttää useammissa aliohjelmissa, tehdään siitä globaali muuttuja. Muuttujan aakkosnumeerinen tieto esitetään lainausmerkeissä, numerot ja totuusarvot ilman lainausmerkkejä.

Vaikka tässä dokumentissa ei korostetusti käytetäkään eri tietotyyppejä, on niiden lyhyt esittely kuitenkin paikallaan.

Tietotyyppi	*VB-vastine*	*Esimerkki*	*Sisältö ja suuruusluokka*
Tavu	*Byte*	*tavu*	*Lyhyt kokonaisluku 0...255*
Kokonaisluku	*Integer*	*luku%*	*-32768...+32767*
Reaaliluku	*Single*	*desim!*	*32 bittinen tarkkuus eli 10E-45...10E38*
Kaksoistarkkuus	*Double*	*tarkka#*	*64 bittinen tarkkuus eli 10E-324...10E308*
Merkkijono	*String*	*jono$*	*2 Gb aakkosnumeerisia merkkejä*
Totuusarvo	*Boolean*	*onjo*	*joko tosi tai epätosi eli True tai False*
Variantti	*Variant*	*arvo*	*voi sisältää mitä tahansa aiempia tai objektin*

Muuttujan esittely voi tapahtua kahdella tavalla. Seuraavassa määritellään kokonaislukumuuttuja, johon tarkoitus sijoittaa koulun oppilaiden lukumäärä kahdella tavalla. Yleensä ohjelmoija ottaa käyttöönsä vain toisen tavan. Muista, että saman nimisiä muuttujia voi käyttää eri aliohjelmissa.

```
Dim OppilasMaara As Integer
tai
Dim OppilasMaara%
```

Visual Basic -ympäristössä ei välttämättä tarvitse määritellä muuttujia ennen niiden käyttöönottoa (Option Implicit). Tämä on kuitenkin jossain määrin vaarallista, koska muuttujanimen voi muistaa tai kirjoittaa väärin. Jotta aina haluttu oikea tieto löytyisi muuttujista, kannattaa muuttujien määrittelystä tehdä pakollinen (Option Explicit) seuraavalla valinnalla

Tools | Options... | Editor | Require Variable Declaration

Muutos astuu voimaan seuraavaa uutta projektia aloitettaessa.

Nimeämisessä kannattaa olla systemaattinen; käytä ohjelmasta toiseen joko kokonaan suomea tai englantia, ei sekaisin. Tämän kirjan harjoituksissa käytetään suomea. On muistettava, ettei nimessä saa olla välilyöntiä, erikois- tai

skandinaavisia merkkejä. Vaikka VB-syntaksi ei sitä edellytäkään, kannattanee useasta sanasta muodostetun nimen jokainen sana alkaa isolla kirjaimella.

Esimerkki1:X=X+1

Ohjelmassa nappulan opaste muutetaan jokaisella painalluskerralla yhtä numeroa suuremmaksi.

- ***ohjelmointiympäristöön tutustuminen***
- ***muuttujan esittely***
- ***muuttujan arvon muuttaminen***
- ***objektin sijoittaminen lomakkeelle***
- ***objektin ominaisuuden muuttaminen ohjelmallisesti***
- ***numeerisen ja aakkosnumeerisen tiedon yhteenlasku***
- ***sovelluksen tallentaminen***

1. Aloita uusi sovellus (File | New Project | Standard EXE - saman valinnan saat CTRl-N-näppäilyllä)

2. Luo lomakkeelle Windows-painike (CommandButton)

3. Muuta painikkeen seuraavia ominaisuuksia (Käytä oikealla näkyvää Ominaisuudet ikkunaa)

.Name	***Nappula***
.BackColor	***DeskTop***
.Caption	***"0"***
.Style	***Graphical***

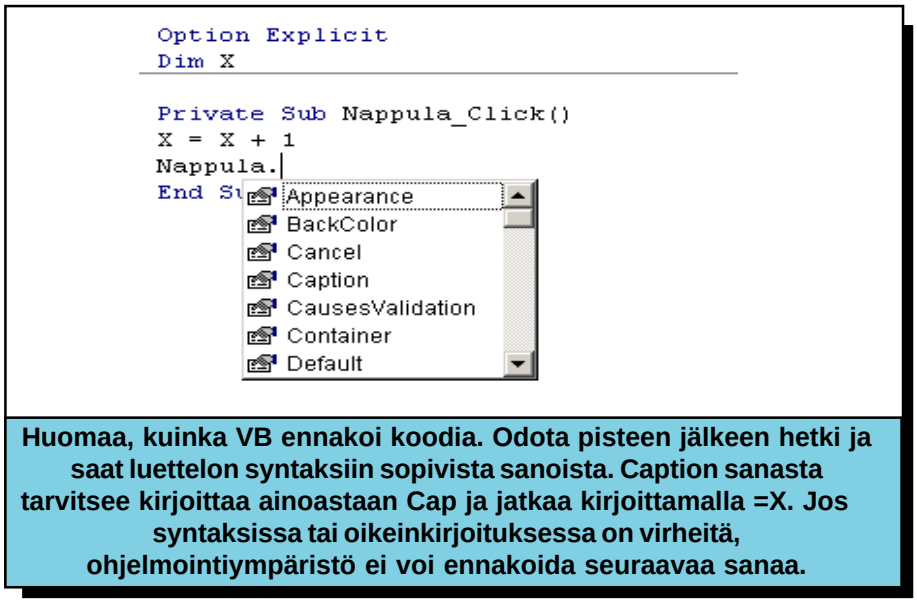
Huomaa, ettei taustaväri muutu ennen kuin nappulasta on tehty tyyliltään graafinen.

4. Määrittele variantti muuttuja X globaaliksi eli aktivoi painike lomakkeella ja paina F7, jolla pääset koodieditoriin. Painikkeen oltua aktiivisena VB-ympäristö tekee nappulalle sen oletustapahtuma-aliohjelman Nappula_Click, jota jatketaan seuraavassa kohdassa.
Kirjoita nyt kuitenkin Option Explicit -määrityksen alle Dim X, joka määrittelee uuden variantin muuttujan X.

5. Jatka koodaamista nappulan osalta ja kirjoita seuraavat rivit

```
X=X+1
Nappula.Caption=X
```

6. Aja sovelluksesi painamalla F5.



Huomaa, kuinka VB ennakoi koodia. Odota pisteen jälkeen hetki ja saat luettelon syntaksiin sopivista sanoista. Caption sanasta tarvitsee kirjoittaa ainoastaan Cap ja jatkaa kirjoittamalla =X. Jos syntaksissa tai oikeinkirjoituksessa on virheitä, ohjelmointiympäristö ei voi ennakoida seuraavaa sanaa.

Nappulasta painettaessa sen kehote kasvaa aina yhdellä. Ohjelmaan kirjoitettu rivi Dim X tarkoittaa, että ohjelma varaa keskusmuistista paikan tiedolle jota tässä kutsutaan X:ksi. Matemaatikon silmissä epäloogiselta näyttävä lause X=X+1 voidaan suomentaa sanoilla X saa arvoksi X+1, jolloin se tuntuu mielekkäämmältä. Kun X:n entinen arvo on korvautunut uudella yhtä suuremmalla arvolla, annetaan nappulan kehotteelle arvoksi X:n sisältämä arvo.

7. Lopeta ohjelma painamalla ALT-F4.

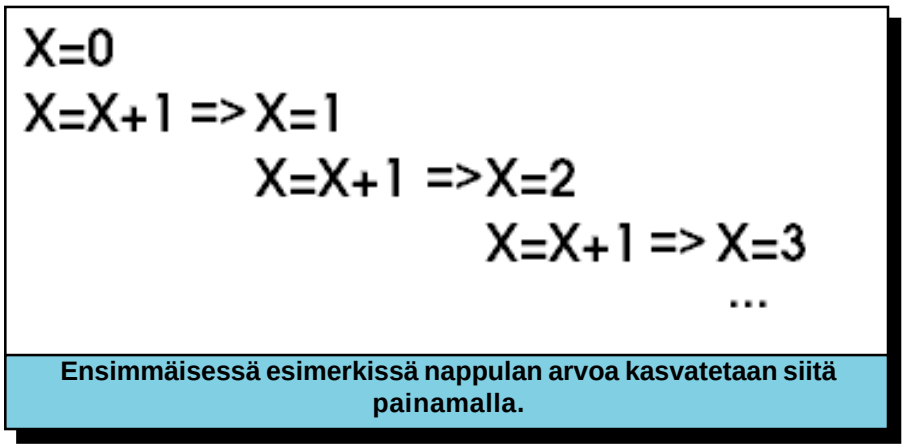
8. Tallenna projekti (File | Save Project)
Huomaa, että aluksi tallennat lomakkeen. Anna sille nimeksi X+1.FRM ja projektille nimeksi X=X+1.VBP. Tiedostotarkenteet .FRM ja .VBP tulevat sanoista Form ja Visual Basic Project, eikä niitä tarvitse itse välttämättä kirjoittaa. Lomakkeiden nimeäminen tulee tärkeäksi, kun sovelluksessa (so. projektissa) joudutaan käyttämään useampia lomakkeita.

Huomioita ohjelmasta:

X:n täytyy olla globaali muuttuja. Jos se olisi lokaali (Nappula_Click -aliohjelmassa määritelty), numerolla kasvanut arvo tuhoutuisi aina aliohjelman suorituksen jälkeen, eikä nappulan arvo koskaan kasvaisi yhtä suuremmaksi.

Saman ohjelman voi tehdä ilman muuttujaakin, muuttamalla suoraan painikkeen kehotteen arvoa
Nappula.Caption = Nappula.Caption + 1
Monista muista ohjelmointikielistä poiketen VB sallii aakkosnumeerisen ja numeerisen tiedon yhteenlaskun.

Windows-painike Nappula voitaisiin nimetä toisinkin. Tässä kirjassa ei käytetä ns. unkarilaista nimeämistä, jossa kolme ensimmäistä merkkiä tulee Windows-objektista ja loppuosa objektin käyttötarkoituksesta. Esimerkin nappulan nimi voisi tällöin olla joko cmdKasvata tai cmdGrow. Kaikilla objekteilla on nimi vaikkei niitä erikseen nimetäkään - ensimmäisen painikkeen oletusnimi on Command1, lomakkeen Form1 jne.



Ensimmäisessä esimerkissä nappulan arvoa kasvatetaan siitä painamalla.

3. INPUT JA OUTPUT

Tässä luvussa ei suinkaan käsitellä kaikkia IO-toimintoja, vaan ainoastaan esitellään yksinkertaiset tavat saada käyttäjältä syötettä (input) ja toisaalta tulostaa tieto (output) kahdella tavalla luomatta Windows-objekteja lomakkeelle. Voimme muuttaa aiempaa ohjelmaa siten, että käyttäjältä kysytään aina nappulan painalluksen jälkeen kuinka paljon arvoa kasvatetaan ja vasta sitten arvo muuttuu. InputBox-funktion ensimmäinen parametri on syöttölaatikon kehote ja toinen ikkunapalkissa näkyvä otsikko - muut parametrit jätetään tyhjiksi.

Esimerkki2:InputBox

Ohjelmassa nappulan kehote muutetaan InputBoxista saadulla tiedolla.

- ***aakkosnumeerisen tiedon yhdistäminen***
- ***InputBox eli syötteen kysyminen käyttäjältä***
- ***Val-muunnos***
- ***sovelluksen tallentaminen uuteen paikkaan***

1. Avaa aiempi sovellus. Jos VB ei ole käynnissä, voi sovelluksen avat resurssienhallinnasta projektin (.VBP) kohdalla kaksoisnäpäyttämällä. Lomakkeen saat aktiiviseksi Projektinhallinnasta oikealta aluksi kaksoisnäpäyttämällä Forms-kansion auki ja sitten Form1(X+1.frm)

2. Aktivoi painike ja siirry sen koodiin (F7). Toinen tapa aloittaa objektin tapahtuma-aliohjelman koodaus on sen kaksoisnäpsäyttäminen lomakkeella.

3. Pyyhi aiempi asetuslause X=X+1.

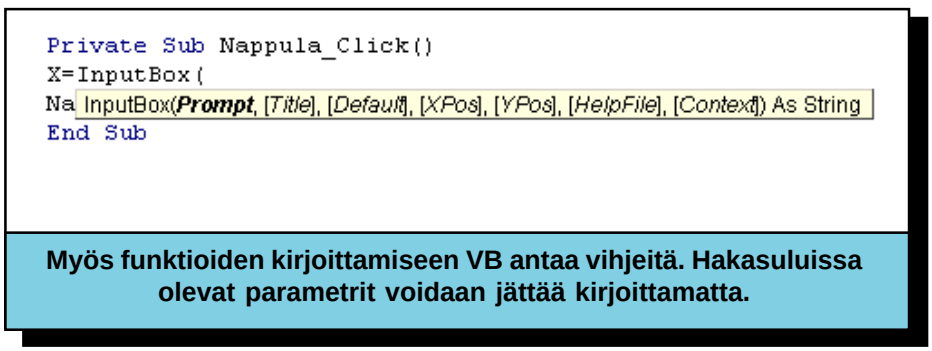
4. Kirjoita seuraavat rivit.

```
X = InputBox("Anna arvo", "Painikkeen arvon kasvatus")
Nappula.Caption = Nappula.Caption + X
```

5. Tallenna aluksi lomake uuteen paikkaan File | Save X+1.frm As - valitse levykeasema uudeksi tallennuspaikaksi ja tallenna nimeä muuttamatta

6. Lomakkeen tallentamisen jälkeen valitse File | Save Project As - ja tallenna levykkeelle antamalla nimeksi InputBox

7. Testaa ohjelma ja ylläty



Myös funktioiden kirjoittamiseen VB antaa vihjeitä. Hakasuluissa olevat parametrit voidaan jättää kirjoittamatta.

Huomioita ohjelmasta

Nyt nappulan arvo ei kasvakaan edellisen harjoituksen tavoin. Vaikka vastaukseksi voi antaa numeron, InputBoxista saatava arvo on aina aakkosnumeerinen. Kahden aakkosnumeerisen arvon yhteenlasku kasvattaa merkkijonoa. Aakkosnumeerisen tiedon yhdistämisessä kannattaa selkeyden takia käyttää kuitenkin &-operaattoria.

"Aku" & " " & "Ankka" mieluummin kuin "Aku" + " " + "Ankka"

Jos uusi ohjelma halutaan toiminnaltaan aiemman kaltaiseksi, tulee X muuttaa numeeriseksi tiedoksi Val-muunnoksella. Tällöin painikkeen arvo kasvaa ainoastaan, jos InputBoxissa on annettu muodoltaan oikeita numeroarvoja. Oikea desimaalierotin on piste, muutoin Val-muunnos katkaisee luvun ennen pilkkua. Ongelmalliset kohdat koodissa kannattaa kommentoida. Visual Basicissa kommentti aloitetaan heittomerkillä ja sen vaikutus ulottuu rivin loppuun saakka.

'Muutetaan InputBoxin arvo yhteenlaskua varten numeeriseksi Val-funktiolla
X=Val(InputBox("Anna arvo", "Painikkeen arvon kasvatus"))

Kun sovellus joudutaan tallentamaan uuteen paikkaan, muista aina tallentaa lomake ensin. Jos tallennat vain projektin, VB olettaa että lomake halutaan säilyttää aiemmassa paikassa.

Esimerkki3: MsgBox

Ohjelmassa tulostetaan sovelluksen nimi kahdella eri tavalla.

- *MsgBox tulostaa tiedon viesti-ikkunaan, jonka nappulat on ohjelmoijan päätettävissä*
- *Print tulostaa suoraan lomakkeelle*
- *ajettavan .EXE-version tekeminen*

1. Aloita uusi projekti

2. Muuta lomakkeen ominaisuuksia

.Name	*Lomake*
.Caption	*TulostusDemo*

3. Luo Windows-painike

.Name	*Nappula*
.Caption	*Tulosta*

4. Tee painikkeeseen seuraava aliohjelma

Viesti-ikkunan nappulat voidaan valita kätevästi luettelosta.

```
Private Sub Nappula_Click()
Dim Vastaus
  Vastaus = MsgBox("Tulosta?", vbExclamation + vbYesNo)
  If Vastaus = vbYes Then
    Print Lomake.Caption
  End If
End Sub
```

5. Tallenna aiempien ohjeiden mukaan. Anna projektille nimi MSGBOX.

6. Ajettavan ohjelman, jossa ei ole VB:n sorsakoodia saat valinnalla File | Make MsgBox.exe. Tätä tiedostoa voi ajaa joko kokonaan itsenäisesti tai suoritettavaksi VB:n RunTime-ympäristössä (koneessa oltava MSVBVM60.DLL).

Huomioita ohjelmasta

MsgBox on funktio, jonka arvo on käytettävä jos toinen parametri on määritelty. Koska funktion muotoparametri on tässä numerotietoa, voidaan niitä myös kutsua useampia yhteenlaskun lopputuloksena. vbExclamination muuttaa ikkunan muotoa siten, että siinä näkyy huutomerkki keltaisella pohjalla. Sama lyhyemmin esitettynä olisi

```
Private Sub Nappula_Click()
  If MsgBox("Tulosta?", vbExclamation + vbYesNo) =vbYes Then
    Print Lomake.Caption
  End If
End Sub
```

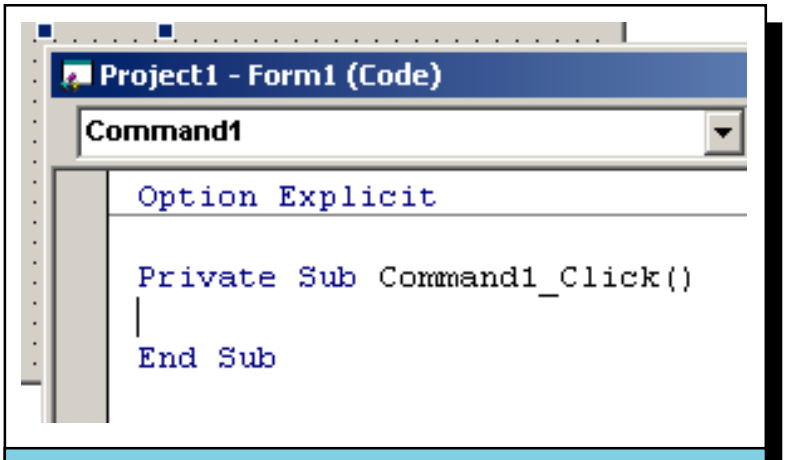
Ohjelman kulkuun saadaan haarakohta If-rakenteella. Ehdon toteutuminen tai toteutumatta jääminen määrittelee kuinka ohjelmassa edetään. Tässä ohjelman otsikko tulostuu lomakkeelle ainoastaan painamalla Yes-nappulaa. Print-komento on kovin epäwindowsmainen tapa tulostaa tietoa, joten sitä ei liiemmin jatkossa käytetä.

4. TAVALLISIMPIEN KOMPONENTTIEN YLEISIMMÄT OMINAISUUDET, METODIT JA TAPAHTUMAT

Algoritmien ratkomisen ohella Windows-ohjelmissa huomio kiinnittyy ohjelmavuon suunnittelun sijaan kontrollirakenteisiin kohdistuviin tapahtumiin, jotka ohjelmaa käytettäessä muuttavat ohjelman kulkua. VB-ympäristössä komponenttien ominaisuuksia voidaan muuttaa joko Ominaisuudet-ikkunasta tai ohjelmallisesti. Metodit ovat komponenttien käyttäytymistä muuttavia aliohjelmia. Tässä kirjassa komponenteista käytetään alkuperäisiä englantilaisia ja suomalaisia termejä sekaisin. Datakomponenttiin liittyvät asiat ja Drag&Drop-toiminnot käsitellään myöhemmin omissa luvuissaan.

Ominaisuudet

.Alignment	***Tekstin sijoittelu tekstialueen vasempaan reunaan, keskelle tai tasattuna oikeaan reunaan.***
.BackColor	***Taustaväri***
.ForeColor	***Tekstin väri (Frame, Label, TextBox)***
.Caption	***Otsikkoteksti (CommandButton, Form, Frame, Label)***
.Enabled	***Ominaisuus, joka osoittaa onko ko. komponentti aktiivinen vastaanottamaan käyttäjän toimenpiteitä. Epäaktiivisen objektin arvoja voi kuitenkin ohjelmallisesti muuttaa.***
.Height	***Komponentin korkeus***
.Index	***Taulukoitujen komponenttien järjestysluku, jota ei voi ohjelmallisesti muuttaa. Jos kontrollirakenne ei ole taulukon alkio, ei sillä ole myöskään indeksiä.***
.Left	***Välimatka komponentin alustana toimivan rakenteen (lomake, kehys) vasemmasta reunasta Windows- yksiköinä***
.TabIndex	***Komponentin aktivointivuoro näppäimistöllä kontrollirakenteiden välillä siirryttäessä***
.TabStop	***Vain ne komponentit, joiden .TabStop-ominaisuus on asetettu todeksi, voivat aktivoitua näppäimistöltä tabulaattorilla siirryttäessä***
.Tag	***Tag-kenttään ohjelmoija voi jättää omia merkintöjään, joita varten objektin muut ominaisuudet eivät kata***
.Text	***Objektin tekstisisältö (TextBox, ComboBox)***
.Top	***Välimatka komponentin alustana toimivan rakenteen (lomake, kehys) yläreunasta Windows- yksiköinä***
.Width	***Komponentin leveys***
.Visible	***Komponentin näkyvyysominaisuus (True/False)***



Koodieditorin aktivoiduttua raamit aktiivisen objektin oletustapahtumalle ovat jo olemassa.

Tapahtumat

Koodieditorin aktivoinnista saatava oletustapahtuma osuu usein kohdalleen. Tapahtumia voi testata yksinkertaisesti kirjoittamalla vastaavia tapahtuma-aliohjelmia, jotka kukin kertovat mistä tapahtumasta on kyse.

```
Private Sub Combo1_Click()
  MsgBox(“Combo1_Click!”)
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
  MsgBox(“Text1_KeyPress!” & KeyAscii)
End Sub
```

_Change()

Tapahtuu, kun objektin sisältö muuttuu. ComboBoxissa muuttuminen tarkoittaa sen rivien määrän muuttumista ja Click-tapahtuma varsinaista valintaa, joten ComboBoxin oletustapahtuma on ehkä väärin valittu.

_Click()

Kun nappulaa tms. on painettu joko hiirellä (MouseUp) tai näppäimistöltä (KeyUp).

_KeyPress(KeyAscii As Integer)

Näppäimistön jonkin näppäimen painalluksesta aiheutunut tapahtuma. Parametri KeyAscii kertoo mitä näppäintä on painettu sen ASCII-desimaalivastineena. ASCII-koodeista lisää myöhemmin.

_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

Hiirtä painettaessa painikkeen vielä ollessa alhaalla.

_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Hiirtä liikutettaessa. Huomaa, että X ja Y tarkoittavat arvoja komponentin vaikutuspiirissä. Jos koordinaattien tarvitsisi korreloida koko sovelluksen koordinaatteja, tulisi niihin lisätä komponentin Left- ja Top-arvot.

_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

Hiiren painalluksen jälkeen juuri kun painike on vapautunut.

Metodit

Metodit ovat aliohjelmia ja erona ominaisuuksiin tarvittavat tiedot välitetään parametreina, eikä asetuslauseessa.

Refresh	***Komponentin päivittäminen. Edellytyksenä sille, että tiedot muuttuvat näytöllä reaaliaikaisesti.***
SetFocus	***Kohdistimen siirto ko. komponenttiin.***
.ZOrder	***Komponentin sijainti Z-akselilla. 0 tarkoittaa päällimmäistä objektia. Yksinkertaisen animaation voi toteuttaa ohjelmana, joka muuttaa kuvien ZOrder-ominaisuutta ajastimen avulla.***

Form (lomake)

Lomake toimi sovelluksen komponenttien alustana. Lomakkeen tapahtumista sen latautuminen (muuttujien ja komponenttien alustus) ja sulkeutuminen (lupa lopettaa) ovat ohjelmissa huomioitava. Usean lomakkeen sovelluksista myöhemmin.

_Load()

Lomakkeen oletustapahtuma, joka tapahtuu yleensä sovelluksen käynnistyttyä, kun lomake kaikkine komponentteineen on kokonaan valmis.

_Unload(Cancel As Integer)

Kun lomake on valittu suljettavaksi, voidaan Cancel asettaa arvoon False, jolloin sovellusta ei voi lopettaa.

CommandButton (painike) 

Aiemmin jo käytetystä CommandButton-komponentista on muistettava muuttaa sen tyyli (.Style) graafiseksi, jotta ominaisuudet kuten BackColor ja Picture näkyisivät. Oletustapahtuma _Click() lienee Windowsin historian käytetyin.

Label (otsikot) ja TextBox (tekstikenttä) 

Sovelluksen otsikot ja kehotteet tehdään labeleina. Suurin osa kehotteista kirjoitetaan ominaisuuksina, eikä niitä juurikaan ohjelmallisesti muuteta - joten niiden nimeäminenkin on tällöin ajan tuhlausta. Toki virheilmoituksina tms. toimivat kehotteet voivat muuttua ohjelman suorituksen aikana.

Tiedon syöttö tapahtuu tekstikenttiin. Joskus myös otsikkomainen tieto voidaan sijoittaa tekstikenttään, mutta silloin kenttä on lukittu (.Locked). Monirivinen tekstikenttä vaatii Multiline-ominaisuuden ja tarvittaessa vierityspalkit (.ScrollBars). Salasanakentäksi muuttaminen tapahtuu PasswordChar-ominaisuuden määrittelemällä.

CheckBox ja OptionButton 

Käytössä CheckBoxit ja OptionButtonit (=radiopainikkeet) eroavat siten, että pyöreitä radiopainikkeita voi olla samalla alustalla (lomake tai kehys) valittuna vain yksi. Ohjelmallisesti CheckBoxin arvo (.Value) voi olla 0 tai 1, kun OptionButtonin arvo on joko true tai false. Käytännössä kontrollirakenteet koipioidaan taulukoksi ja arvoja testataan toistorakenteessa.

ComboBox ja ListBox (lista) 

Nämä kaksi komponenttia sisältävät listoille tyypillisiä ominaisuuksia. ComboBox toimii valintarakenteena, jossa ylin (näkyvä) rivi on ComboBoxin Text-ominaisuus. Muut rivit kummassakin listassa löytyvät niiden List-ominaisuudesta. Kullakin listan rivillä on rivinumeroa vastaava Index-ominaisuutensa. Rivejä voidaan lisätä AddItem- ja poistaa RemoveItem(rivinumero)-metodilla. ListIndex-ominaisuus kertoo aktivoidun rivin indeksin ja ListCount rivien kokonaismäärän.

Timer (ajastin) ja OLE 

Ajastimen käytössä ominaisuudet Enabled (onko ajastintoiminnot päällä) ja Interval (aikaväli) joudutaan ohjelmissa huomioimaan, kun oletustapahtuma (Timer) osoittaa kellon pyörähtäneen jälleen kerran eteenpäin.

Toisissa sovelluksissa luodut OLE-objektit yhdistetään omiin sovelluksiin CreateLink- ja otetaan käyttöön DoVerb-metodilla.

Esimerkki4: Kuvat

Ohjelmassa arvotaan kaksi noppalukua, näytetään nopat ja lisätään voitto-saraketta, jos silmäluvuksi saadaan seitsemän.

- ***kuvien käyttö***
- ***satunnaisluvut***
- ***laskurimuuttujat***

1. Anna lomakkeen otsikko

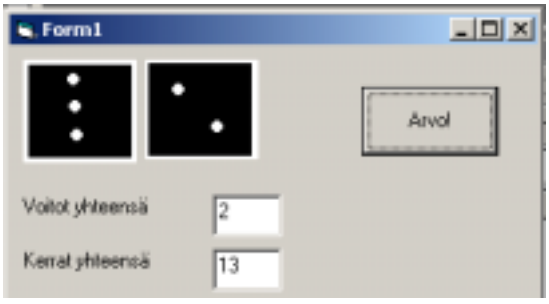
.Caption	***Lucky Seven***

2. Luo ohjelman kaksi kuvaobjektia

.Name	***Noppa1***
.Name	***Noppa2***

3. Tee painike

.Name	***Arvo***
.Caption	***Arvo!***



Pelissä voittaa, jos lukujen summa on 7.

4. Tee otsikko Voitot yhteensä ja Kerrat yhteensä (objekteja ei tarvitse nimetä)

5. Tee otsikoiden viereen tekstilaatikot ja lukitse ne

.Name	***Voitot***
.Locked	***True***
.Text	***(tyhjä)***
.Name	***Kerrat***
.Locked	***True***
.Text	***(tyhjä)***

6. Piirrä kuusi biittikarttakuvaa noppien silmäluvuista 1-6 ja nimeä ne NOPPA1.BMP, NOPPA2.BMP ... NOPPA6.BMP.

Jos internet on käytössäsi, voit etsiä kuvia osoitteesta http://google.fi hakusanalla dice.

7. Kirjoita Arvo-painikkeen koodi

```
Private Sub Arvo_Click()
Dim Arvottu, Summa
Arvottu = Int(Rnd * 6) + 1
Noppa1.Picture = LoadPicture("NOPPA" & Arvottu & ".BMP")
Summa = Arvottu
Arvottu = Int(Rnd * 6) + 1
Noppa2.Picture = LoadPicture("NOPPA" & Arvottu & ".BMP")
Summa = Summa + Arvottu
Kerrat.Text = Val(Kerrat.Text) + 1
If Summa = 7 Then
  Voitot.Text = Val(Voitot.Text) + 1
End If
End Sub
```

8. Tallenna sovellus samaan hakemistoon kuvien kanssa
9. Tee sovelluksesta .EXE-versio

Huomioita ohjelmasta

Ohjelmassa asetuslauseessa tekstikentän sisältö on muutettava numeeriseksi ennen yhteenlaskua. Sen sijaan muutosta toisin päin numerosta tekstiksi (str-funktiolla) ei tarvita. Satunnaislukufunktio palauttaa yhtä pienemmän positiivisen desimaaliluvun, joten se tulee kertoa vaihtoehtojen lukumäärällä, koska int ei pyöristä ylöspäin, vaan palauttaa viimeisen mukaan mahtuvan kokonaisluvun. Yksi lisätään arvottuun kokonaislukuun, ettei nolla tulisi nopan silmäluvuksi. Jos ajat ohjelman useasti peräkkäin, huomaat että samat numerosarjat toistuvat samanlaisina. Jos ohjelmassa halutaan jokaisella suorituskerralla erilaiset numerosarjat, tulee satunnaislukugeneraattorin siemenlukua muuttaa Randomize-aliohjelmalla esim. Form_Load-aliohjelmassa. Kuvien sijainti saattaa aiheuttaa ongelmia, jos testaat ohjelmaa VB-ympäristössä. Lisää kuvien hakemistopolku testausvaiheessa tiedostonimen eteen mieluummin kuin siirrät kuvatiedostot VB:n työhakemistoon. Muista kuitenkin, että .EXE-versio on se, joka yleensä jaetaan muille käyttäjille.

5. BASIC-KIELEN TOISTO- JA EHTORAKENTEITA

Niin kuin aiemmin on jo todettu, on ohjelmointikieli sidottu sen syntaksiinsa. Variantit muuttujat, muuttujien määrittelemättä jättäminen ja aliohjelman parametrien lukumäärän vaihtelu tekee Visual Basicin kieliopiltaan väljemmäksi kuin monet muut ohjelmointikielet. Väljä syntaksi tai ei niin voidaan sanoa, että ohjelmointi alkaa tosimielessä vasta kun ohjelmoija on pystyy hyödyntämään taulukkomaisia rakenteita erilaisissa ehdollisissa toistorakenteissa. Tässä luvussa muutamia VB:n ehto- ja toistorakenteita.

Monipuolinen If-ehtorakenne

Perinteinen ehtorakenne If näyttää Basicissa tällaiselta:

```
If Ehto Then
  Toimenpiteet
End If
```

Jos toimenpiteitä on yksi, voi sen kirjoittaa If-riville, eikä End If-riviä tarvita.

Basicin Case-rakenne tuntuu hiukan väkinäiseltä, eikä sitä tässä käsitellä. Sen sijaan If-rakenteeseen saadaan lisävaihtoehtoja valinnaisilla Else- ja ElseIf-haaroilla. Samalla säästetään End If -rivien määrässä.

Syntaksi

```
If ehto Then
  Toimenpiteet
ElseIf ehto Then
  Toimenpiteet
ElseIf Then
  Toimenpiteet
Else
  Toimenpiteet
End If
```

Esimerkki

```
If Arvo<100 Then
  MsgBox(“Pieni”)
ElseIf Arvo>1000 Then
  MsgBox(“Suuri”)
Else
  MsgBox(“Keskiverto”)
End If
```

For-toisto

For-toisto on usein helpoin tapa toteuttaa silmukoita, joiden lukumäärä on toiston alettaessa ohjelmoijan tiedossa tai muuttujiin tallennettuna. Step-arvoa käyttämällä toiston askellusväli voi olla muutakin kuin yksi ja toiston voi suorittaa myös takaperin.

Syntaksi

```
For LaskuriMuuttuja=Alkuarvo To  Loppuarvo [Step Arvo]
  toimenpiteet
Next Laskurimuuttuja
```

Esimerkki

```
For I=1 to 3       ‘Toistetaan kolme kertaa
  MsgBox(I*I)      ‘Tulostetaan I²  eli 1,4,9
Next I             ‘Palataan toiston alkuun
```

Do While ja Do Until -toistot

Monessa ohjelmointikielessä on kaksi erilaista ehdollista toistorakennetta; toisessa ehtorakenne on lopussa, jolloin toistetaan ainakin yhden kerran ja toisessa ehto on toistosilmukkaan pääsyn ehtona ennen kuin on toistettu yhtään kertaa. VB:ssä ehdollisissa toistoissa käytetään kahta englanninkielistä sanaa: while (niin kauan kuin) ja until (niin kauan kunnes) ja nämä sanat voivat esiintyä joko silmukan aloitusehtona tai lopetusehtona Do Loop-rakenteessa. Seuraavassa esimerkki Until-ehdosta

Syntaksi

```
Do
 toimenpiteet
Loop Until Ehto
```

Esimerkki

```
mones=0
Do
  mones=mones+1
  Print(mones & " pieni elefantti marssi näin")
  Jatketaanko=MsgBox("Jatketaanko?",vbYesNo)
Loop Until Jatketaanko=vbNo
```

Tosin edellisessä lopetusehto voisi olla Loop Until InputBox("Jatketaanko?,vbYesNo)=vbNo. Ehtorakenteissa ja ehdollisissa toistorakenteissa käytettävät vertailuoperaattorit ovat seuraavat:

<	***pienempi kuin***
<=	***pienempi tai yhtä suuri kuin***
=	***yhtä suuri kuin***
<>	***erisuuri kuin***
>	***suurempi kuin***
>=	***suurempi tai yhtä suuri kuin***

Operaattorien molemmin puolin voi olla muuttujia, vakioarvoja tai laskukaavoja ja erilaisia ehtoja voidaan yhdistää AND- ja OR-rakentein. Ehtolauseke, joka voidaan ohjelmaa testattaessa joutua tulostamaan vaikkapa muodossa MsgBox(A=B), antaa tulokseksi arvon True tai False.

6. ALIOHJELMAT

Suomen kielessä puhutaan lausealiohjelmista (proseduureista) ja funktio-aliohjelmista. Aliohjelmien ero on siinä, että funktioaliohjelma palauttaa jonkin arvon, joka pitää sitä kutsuttaessa myös käyttää. Muuten lauseet aliohjelmien sisällä voivat olla toistensa kaltaisia. Arvon palauttaminen tarkoittaa yleensä laskun lopputulosta. Usein aliohjelmiin myös viedään tietoa parametreina. Näitä tietoja käytetään aliohjelman lasku- tms. lausekkeissa.

Lausealiohjelman syntaksi

```
Sub Aliohjelma(valinnaiset parametrit)
 toimenpiteet
End Sub
```

Funktioaliohjelman syntaksi

```
Function Aliohjelma(valinnaiset parametrit)
 toimenpiteet
 asetuslause, jossa funktio saa arvonsa
End Function
```

Yksinkertaisimmillaan funktio voi olla vaikkapa tällainen

```
Function Palauta
  Palauta=”Aku Ankka”
End Function
```

ja koska sen arvo pitää käyttää, sitä voisi kutsua joko asetuslauseessa tai toisesta aliohjelmasta

```
MJono=Palauta
MsgBox(Palauta)
```

Tässä esimerkkejä tyypillisistä funktioista, joihin arvoja viedään parametreina ja jotka palauttavat funktioon viedyistä arvoista laskutoimituksen avulla lopputuloksen. Aliohjelman sisällä käytetään funktion määrittelyssä kirjoitettuja parametrinimiä - ei funktion kutsussa määriteltyjä arvoja

```
Function Tulo(Kertoja,Kerrottava)
  Tulo=Kertoja*Kerrottava
End Function

Function Euribor(Laina, Aika, Prosentti)
  Euribor = Laina * Aika * Prosentti / 100 / 12
End Function
```

Aliohjelman kutsussa halutut arvot viedään joko vakioina tai muuttujina (tai laskulauseina) funktioon. Pitää muistaa, että funktion arvo on “kulutettava” sitä kutsuttaessa.

```
Luku=3
MsgBox(Tulo(Luku,5))
MsgBox (Euribor(Val(Text1.Text), 36, Val(Text2.Text)))
```

Esimerkki5:Piiri ja ala

Ohjelma laskee ympyrän piirin ja alan käyttäjän antaman säteen perusteella

- ***funktiot***
- ***parametrit***
- ***ENTERin käyttö ohjelmissa***
- ***rivinvaihto tekstin joukossa***

(ASCII = American Standard Code for Information Interchange)
(also see Related Links below)

Decimal	Octal	Hex	Binary	Value	
000	000	000	00000000	NUL	(Null char.)
001	001	001	00000001	SOH	(Start of Header)
002	002	002	00000010	STX	(Start of Text)
003	003	003	00000011	ETX	(End of Text)
004	004	004	00000100	EOT	(End of Transmission)
005	005	005	00000101	ENQ	(Enquiry)
006	006	006	00000110	ACK	(Acknowledgment)
007	007	007	00000111	BEL	(Bell)
008	010	008	00001000	BS	(Backspace)
009	011	009	00001001	HT	(Horizontal Tab)
010	012	00A	00001010	LF	(Line Feed)
011	013	00B	00001011	VT	(Vertical Tab)
012	014	00C	00001100	FF	(Form Feed)
013	015	00D	00001101	CR	(Carriage Return)

ENTER-näppäimestä välittyy KeyPress-aliohjelmaan ASCII koodi desimaalinumero 13. Tekstitiedostoissa ENTERiä vastaa CR-LF.

1. Aloita uusi sovellus; lomakkeelle otsikoksi Ympyrän piiri ja ala

2. Tee kolme kehotetta allekkain

.Caption	***Kirjoita ympyrän säde***
.Caption	***Piiri***
.Caption	***Ala***

3. Tee kolme tekstikenttää allekkain

.Name	***r***
.Name	***txtPiiri***
.Name	***txtAla***

4. Valitse lomake aktiiviseksi ja kirjoita sen Load-tapahtuma-aliohjelmaan

```
Private Sub Form_Load()
  Kehote.Caption = Kehote.Caption & vbCrLf & “ja paina <ENTER>”
End Sub
```

5. Kirjoita piirin ja alan laskevat funktiot Form_Load-aliohjelman alapuolelle tyhjään tilaan.

```
Function Piiri(sade)
  Piiri = 2 * 3.14 * sade
End Function

Function Ala(sade)
  Ala = 3.14 * sade * sade
End Function
```

6. Valitse tekstikenttä r aktiiviseksi ja kirjoita sen KeyPress-tapahtuma-aliohjelma. Huomaa, että oletusaliohjelma Change-aliohjelma pitää muuttaa koodieditorin oikeasta yläreunasta oikeaksi.

```
Private Sub r_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 And IsNumeric(r.Text) Then
    txtPiiri.Text = Piiri(r.Text)
    txtAla.Text = Ala(r.Text)
  End If
End Sub
```

Huomioita ohjelmasta

Rivinvaihto saadaan tekstin joukkoon lisäämällä basic-vakio vbCrLf. Basicin vakiot ovat aina vb-alkuisia - aiemmin on käytetty MsgBoxin nappuloista vakioarvoa - myös useat värit voidaan kirjoittaa vakiona, kuten vbRed. Tässä Cr tulee sanasta Carriage Return eli vaunun palautus (kirjoituskonetermi) ja Lf sanasta LineFeed eli rivinvaihto. Vastaavat Ascii-koodit ovat 13 ja 10.

Näppäimistöltä voidaan testata hiiren tapaan erilaisia tapahtumia. KeyPress-aliohjelman parametri on painetun näppäimen Ascii-koodi. Numero 13 vastaa Enterin painallusta. Ohjelmassa edetään siis vain jos sekä painettu näppäin on Enter ja tekstikentän r sisältö on numeroksi sopivaa.

7. TAULUKOT

Tärkein elementti, jonka avulla useat tietokoneohjelmat ovat yleensäkään ohjelmoitavissa (ainakin mielekkäästi), on taulukot. On yksiulotteisia (vektoreita), kaksiulotteisia (matriiseja) ja moniulotteisia taulukoita. Taulukoiden sisältönä on keskenään samankaltaisia tietoja; numeroita, merkkijonoja ja muitakin rakenteita. Kaikkia taulukon alkioita kutsutaan samalla nimellä - ne eroavat kuitenkin toisistaan järjestyslukunsa eli indeksinsä mukaan. Indeksi on taulukon alkion perässä suluissa oleva numero, joka ohjelmissa on yleensä toistorakenteen laskurimuuttuja. VB:n taulukoiden indeksit alkavat oletusarvoisesti nollasta.

Suurin hyöty taulukoiduista tiedoista saadaan toistorakenteissa. Koodissa ei tällöin jouduta kirjoittamaan jokaista alkiota varten omaa koodiansa, vaan samanlaisena toistuvat asiat kohdistuvat vuorollaan jokaiseen alkioon. Tietenkin erityistapaukset tulee huomioida tapauskohtaisesti.

Esimerkkejä taulukoista on loputtomasti. Esimerkiksi luokassa on paljon oppilaita. Jos oppilaiden nimet halutaan johonkin muistiin, ei kannata tuhlata aikaa kirjoittamalla jokaisen oppilaan nimeä varten oma muuttujansa ja koodata lause, jossa tietoa kysytään. Oikea tapa on tehdä oppilaista taulukko ja kysyä nimet toistorakenteessa.

```
Dim Oppilas(36)         ‘Taulukossa on 36 alkiota - tässä nimeä
Dim Indeksi

For Indeksi=0 To 35     ‘Tässä toistetaan 36 kertaa
  Oppilas(Indeksi)=InputBox(“Kirjoita oppilaan nimi”)
Next Indeksi
```

Esimerkki6: Puhelinmuistio

Ohjelmassa luodaan kaksi taulukkoa. Toinen sisältää 10 nimeä ja toinen nimiä vastaavat puhelinnumerot samalla indeksillä. Numerotiedot voidaan hakea joko nimen tai indeksin avulla.

- ***yksiulotteiset taulukot***
- ***Windowsin valikot***

1. Anna sovellukselle otsikoksi P U H E L I N M U I S T I O

2. Tee ohjelman valikot Tools Menu Editor -valinnalla (CTRL-E). Tässä ohjelmassa on kolme päävalintaa Syötä, Hae ja Lopeta. Lisäksi Hae-valinnalla on kaksi alavalintaa. Valinnat kannattaa tehdä kaikki järjestyksessä allekkain. Next aloittaa aina uuden valinnan. Valinnat Nimen perusteella ja Indeksin avulla muutetaan alavalinnoiksi menueditorin nuolella oikealle. &-merkillä valitaan se merkki joka käynnistää valinnan näppäimistöltä. Muista aina nimetä valintasi tai et pääse editorista pois.

Caption	*&Syötä*
Name	*Tiedot*
Caption	*&Hae*
Name	*Hae*
Caption	*&Nimen perusteella*
Name	*Nimihaku*
Caption	*&Indeksin avulla*
Name	*Indeksihaku*
Caption	*&Lopeta*
Name	*Lopeta*

3. Määrittele ohjelmakoodin yläosaan Option Explicitin alapuolelle käytettävät (globaalit) taulukot Nimi ja Numero ja muuttuja Max.

```
Dim Nimi(10), Numero(10), Max
```

4. Tee lomakkeen Load-aliohjelma

```
Private Sub Form_Load()
  Max = 0
End Sub
```

4. Tee Syötä-valinnalle oma aliohjelmansa. Valikkojen tapahtuma-aliohjelmat saavat saman nimen kuin mitä olit aiemmin määritellyt menueditorissa. Aliohjelman kirjoittamiseen pääset yksinkertaisesti valitsemalla valinnan lomakkeelta.

```
Private Sub Tiedot_Click()
  If Max < 10 Then
    Nimi(Max) = InputBox(“Anna nimi”)
    Numero(Max) = InputBox(“Anna puhelinnumero”)
    Max = Max + 1
  End If
End Sub
```

5. Tee aliohjelma Hae nimen perusteella -valinnalle.

```
Private Sub Nimihaku_Click()
Dim Avain, I
Avain = InputBox(“Anna hakuavain”)
For I = 0 To Max - 1
  If Avain = Nimi(I) Then
    MsgBox (Nimi(I) & “:” & Numero(I))
  End If
Next I
End Sub
```

6. Tee aliohjelma indeksihaulle

```
Private Sub Indeksihaku_Click()
Dim I
I = InputBox(“Anna haettavan tiedon indeksi”)
 If I < 10 Then
   MsgBox (Nimi(I) & “:” & Numero(I))
 End If
End Sub
```

7. Tee Lopeta-aliohjelma

```
Private Sub Lopeta_Click()
  End
End Sub
```

Huomioita ohjelmasta

Huomaa, että taulukoiden indeksointi alkaa nollasta ellei taulukkoa esiteltäessä toisin määritellä. Nimihaussa toistorakenteessa verrataan vuorollaan kutakin Nimi-taulukon alkiota hakuavaimeen. Suuremmissa taulukoissa toisto pitäisi toteuttaa siten, että se loppuu kun oikea tieto on löytynyt. Toisaalta ohjelmaa voisi kehittää siten, että tulostettaisiin kaikki ne nimet, joissa hakuavain on osana nimeä - tästä enemmän seuraavassa luvussa.

8. TIEDON MUOTOILU JA MUOKKAAMINEN

Tässä lyhyesti esimerkkejä tiedon käsittelystä.

Merkkijonon pituus löydetään funktiolla Len

```
  MsgBox(Len(Text1.Text))
```

Kun halutaan kopioida merkkejä vasemmalta merkkijonosta (Left)...

```
  MsgBox(Left(Text1.Text,3))
```

...merkkejä oikealta (Right)

```
  MsgBox(Right(Text1.Text,5))
```

Jos tieto on kirjoitettu muodossa Etunimi Sukunimi, voidaan sukunimi tulostaa etsimällä välilyönnin paikka (InStr)

```
MsgBox (Mid(Text1.Text, InStr(Text1.Text, " ")+1))
```

Mid-funktio toimii tässä Rightin tavoin, koska valinnainen pituusparametri on jätetty käyttämättä.

Edellisen luvun puhelinmuistiossa InStr-funktiota voitaisiin hyödyntää ns. osa-avainhaussa seuraavalla tavalla

```
For I=0 to Max-1
  If InStr(Nimi(I),Avain)>0 Then MsgBox (Numero(I))
Next I
```

Numeroiden muotoilussa käytetään Format-funktiota. Esimerkissä tulostetaan luku kaksidesimaalisena.

```
MsgBox (Format(Val(Text1.Text), "######0.00"))
```

Päiväyksen muotoiluesimerkkejä

```
MsgBox (Format(Now(), "dddd,dd. mmmm yyyy"))
MsgBox (Format(#6/5/2003#, "dddd,dd. mmmm yyyy"))
```

Huomaa, että päiväys kirjoitetaan jenkkimuodossa (#kk/pp/vvvv#) ja esimerkki palauttaa kesäkuun viidennen päivän tiedot.

Esimerkki7:Kassademo

Ohjelmassa viedään InputBoxilla kerätyt tiedot ListBox-elementtiin. Syöttäminen loppuu, kun annettu tuoterivi on tyhjä.

- ***aakkosnumeerisen tiedon pilkkominen taulukoksi (Split)***
- ***listaobjektin käsittely***

1. Tee sovellukseen listaelementti ja muuta sen fontti Courieriksi. Yleensä eri merkit ovat leveyssunnassa erikokoisia (w,i). Courier on tyyliltään sellainen, että jokainen merkeistä vie yhtä paljon tilaa.

.Name	***Lista***
.Font	***Courier 12***

2. Tee globaali muuttuja KokHinta

3. Tee oma funktio, joka lisää merkkijonon loppuun halutun määrän välilyöntejä

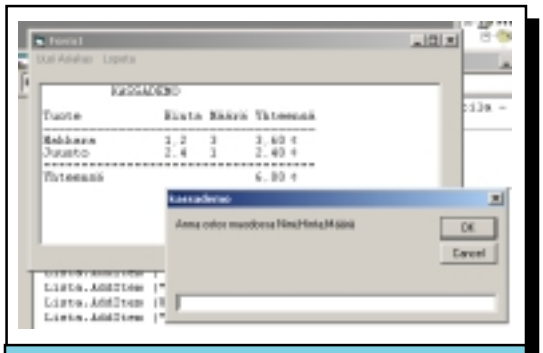
```
Function Uusijono(alkujono, tila)
  Uusijono = alkujono + Left("                          ", tila - Len(alkujono))
End Function
```

4. Tee päävalinta Uusi Asiakas, johon kirjoitetaan seuraava koodi

```
Private Sub Uusi_Click()
Dim Rivi, Ostos, RiviHinta
KokHinta = 0
Lista.Clear
Lista.AddItem ("         KASSADEMO")
Lista.AddItem ("")
Lista.AddItem (Uusijono("Tuote", 16) & Uusijono("Hinta", 6) & &_
Uusijono("Määrä", 6) & "Yhteensä")
Lista.AddItem ("————————————————")
Lista.AddItem ("==================================")
Lista.AddItem (Uusijono("Yhteensä", 28) & &_
Format(KokHinta, "#######0.00 •"))
Lista.Refresh

Do
  Rivi = InputBox("Anna ostos muodossa Nimi;Hinta;Määrä")
  If Rivi <> "" Then
    Lista.RemoveItem (Lista.ListCount - 1)
    Lista.RemoveItem (Lista.ListCount - 1)
    Ostos = Split(Rivi, ";")
    RiviHinta = Format(Ostos(1) * Ostos(2), "######0.00 •")
    KokHinta = KokHinta + RiviHinta
    Lista.AddItem (Uusijono(Ostos(0), 16)) & &_
    Uusijono(Ostos(1), 6) & Uusijono(Ostos(2), 6) & &_
    Uusijono(RiviHinta, 10)
    Lista.AddItem ("==================================")
    Lista.AddItem (Uusijono("Yhteensä", 28) & &_
    Format(Round(KokHinta * 20) / 20, "#######0.00 •"))
  End If
Loop Until Rivi = ""
End Sub
```

5. Tee Lopeta-valinnan aliohjelma

```
Private Sub Lopeta_Click()
  End
End Sub
```



Kassademo.

Huomioita ohjelmasta

Listaan lisättyjä alkioita poistetaan indeksin avulla. Listan viimeisen alkion indeksi on lista alkioiden määrä (ListCount) vähennettynä yhdellä. Tässä ohjelmassa viimeinen rivi poistetaan, koska se muuttuu jokaisen ostoksen jälkeen ja alleviivaus otetaan uuden ostoksen tieltä. Toki AddItem-metodissa on valinnainen indeksi, jolla uusi rivi voitaisiin lisätä keskelle listaa.

Ohjelmassa on jouduttu merkkejä &_ käyttämällä jatkamaan ylipitkä koodi seuraavalta riviltä. Huomaa Enterin painalluksella saatu tyhjä syöte, jossa on peräkkäin kaksi lainausmerkkiä. Samaa tapaa käytetään myös tyhjän rivin lisäämiseksi listaan.

Erotinmerkki(-jono), jolla Split-funktiossa merkkijono jaetaan yksiulotteiseksi taulukoksi, on tässä tarkoituksella valittu puolipisteeksi. Välilyönti erottimena estäisi välilyönnin käytön tuotteen nimessä.

Lopussa ostoksen kokonaishinta pyöristetään lähimpään viiteen senttiin. Ajatuksena on paloitetella summa aluksi viiden sentin osiin so. kahdeskymmenesosa eurosta, pyöristää luku ja palauttaa takaisin oikeaan suuruusluokkaan.

9. KUPLALAJITTELU

Edellisessä harjoituksessa käytettiin listaobjektia, jonka ominaisuuksiin kuuluu myös lajittelu suuruusjärjestykseen (List.Sorted). Lajittelualgoritmien tunteminen on ohjelmoinnin yleistietoa, jota ei välttämättä tarvitse osata ulkoa mutta jonka idea tulisi olla mielessä.

Tässä otetaan esimerkiksi kuplalajittelu, joka ei ole erilaisista algoritmeista välttämättä nopein - vaan selkeä tiedon lajittelualgoritmi, josta mieleen tulisi jäädä seuraavat asiat

- *tieto on taulukoitava*
- *kaikki tapahtuu kahdessa sisäkkäisessä toistossa*
- *vierekkäisiä taulukon alkioita verrataan toisiinsa*
- *jos alkiot ovat epäjärjestyksessä toinen niistä siirretään apumuuttujaan*
- *alkioiden paikat vaihdetaan kumpaakaan tuhoamatta*
- *toistojen lukumäärä on maksimissaan (n-1)*(n-1), missä n kuvastaa alkioiden lukumäärää*

Esimerkki8:Lotto

Tee ohjelma, joka arpoo seitsemän numeroa väliltä 1-39. Vie luvuista taulukkoon ainoastaan sellaiset, jotka eivät siellä vielä ole ja järjestä luvut lopuksi nousevaan suuruusjärjestykseen.

- *tapahtuma-aliohjelman kutsuminen*
- *kuplalajittelu*
- *vakioiden esittely*

Ohjelman hahmottamisen voisi aloittaa puoliohjelmana (=algoritmi ja rakenteet suomen kielellä). Usein paras lähestymistapa on pilkkoa asiat osiin ja testata osia toisistaan irrallaan - sitä varten ohjelmassa on useampi kuin yksi nappula.

```
TOISTA 7 KERTAA
  TOISTA
   OLETETAAN, ETTÄ ARVOTTU LUKU ONNISTUI
   ARVO LUKU TAULUKKOON LOPPUUN
   TOISTA NIIN MONTA KERTAA KUIN TAULUKOSSA OLI JO AIEMMIN LUKUJA
     VERTAA ARVOTTUA LUKUA TOISTOINDEKSIN OSOITTAMAAN TAULUKON ALKIOON
     JOS LUKU=TAULUKON ALKIO => ONNISTUI EPÄ
  KUNNES ONNISTUI
LOPPU TOISTA (KASVATA INDEKSIÄ)
```

1. Määrittele globaali vakio Max, joka saa arvoksi 7 ja sen alle globaali taulukko Lotto, jossa vakion verran alkioita

```
Const Max=7
Dim Lotto(Max)
```

2. Tee nappula Arvo luvut! ja Tulosta

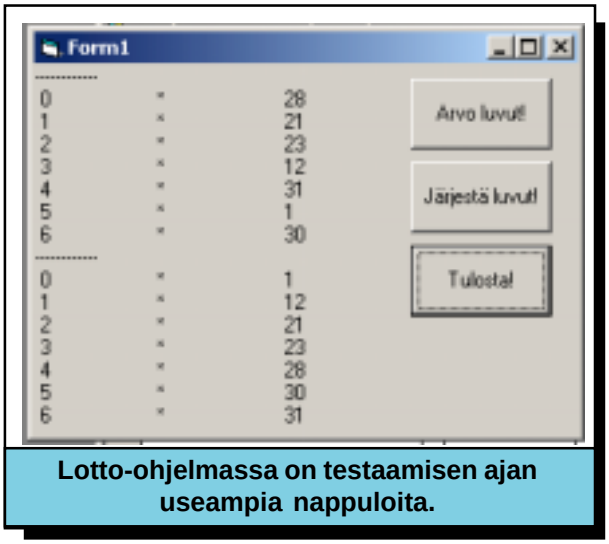
.Name	*Arvo*
.Caption	*Arvo luvut!*
.Name	*Tulosta*
.Caption	*Tulosta*

3. Tee Tulosta-aliohjelma

```
Private Sub Tulosta_Click()
Dim Y
Print "_______"
For Y = 0 To Max - 1
  Print Y, "*", Lotto(Y)
Next Y
End Sub
```

4. Kirjoita Arvo luvut! -nappulaan puoliohjelmaa vastaava basic-koodi ja tulosta testimielessä lomakkeelle Print-komennolla

```
Private Sub Arvo_Click()
Dim Y, I, Onnistui
Cls
For Y = 0 To Max - 1
    Do
      Onnistui = True
      Lotto(Y) = Int(Rnd * 39) + 1
      If Y > 0 Then
        For I = 0 To Y - 1
          If Lotto(Y) = Lotto(I) Then
            Onnistui = False
          End If
        Next I
      End If
    Loop Until Onnistui
  Next Y
  Tulosta_Click
End Sub
```



Lotto-ohjelmassa on testaamisen ajan useampia nappuloita.

4. Tee nappula Järjestä luvut!

.Name Sort
.Caption Järjestä luvut!

5. Kirjoita nappulaan aliohjelma, jossa kahdessa sisäkkäisessä toistossa verrataan taulukon vierekkäisiä alkioita ja vaihdetaan tarvittaessa niiden paikkaa

```
Private Sub Sort_Click()
Dim Apu, Y, X
  For Y = 0 To Max - 2
    For X = 1 To Max - 1
      If Lotto(X) < Lotto(X - 1) Then
        Apu = Lotto(X)
        Lotto(X) = Lotto(X - 1)
        Lotto(X - 1) = Apu
      End If
    Next X
  Next Y
End Sub
```

Huomioita ohjelmasta

Muista, että sisäkkäisessä toistorakenteessa aina ensimmäiseksi alkanut toiston lohko päättyy viimeisenä - sanotaan että "luupit eivät voi mennä ristiin". Kuplalajittelussa toistot voisi lopettaa useassa tapauksessa aikaisemminkin - tällöin ulompi toistorakenne päättyisi kun sisemmässä toistossa ei tapahtuisi ainoatakaan taulukon alkioiden siirtoa.
Huomaa, kuinka Arvo-aliohjelmassa kutsutaan tulostusnappulan tapahtuma-aliohjelmaa.

10. JAKOLASKULLA VEKTORISTA MATRIISIKSI

Joissakin sovelluksissa yksiulotteisen taulukon tiedot esitetään kaksiulotteisessa maailmassa. Saman rivin ja toisaalta saman sarakkeen tiedoilla on tällaisissa sovelluksissa jakolaskun lopputulos yhteisenä tekijänä.

Kokonaislukujako, jolla saadaan taulukon rivi selville, merkitään basic-kielessä kenoviivalla esim.

3 \ 5 ***(lopputulos on nolla)***
6 \ 5 ***(lopputulos on yksi)***

Jakojäännösoperaattori on nimeltään mod - sama jakojäännös kuvastaa samaa saraketta

3 mod 5 ***(lopputulos on 3)***
6 mod 5 ***(lopputulos on 1)***

Esimerkki9:Monivalinta

Tee ohjelma, jossa käydään viisi valintariviä läpi ja ilmoitetaan aina, jos rivin päädyssä on nappula valittuna (Ääripäiden vastaukset kiinnostavat yleensä gallupeissa eniten)

- ***kokonaislukujako ja jakojäännös***
- ***radiopainike***
- ***kontrollirakenteiden kopiointi taulukoksi***

1. Tee viisi kehystä – anna selkeyden vuoksi joka toiselle kehykselle eri taustaväri

2. Tee yksi radiopainike (Optionbutton) ensimmäiselle kehykselle.
.Name ***Valinta***

3. Kopioi radiopainike (CTRL-C ja CTRL-V). Vastaa kyllä, kun VB kysyy tehdäänkö rakenteesta kontrollirakennetaulukko. Jatka kopiointia täyttäen järjestyksessä kaikki viisi riviä viidellä valintapainikkeella.

4. Tee lomakkeenlatausaliohjelmaan koodi, joka antaa radiopainikkeille kehotteet A:sta E:hen.

```
Private Sub Form_Load()
Dim I
For I=0 to 29
  Valinta(I).Caption=Chr(65)+I Mod 5
Next I
End Sub
```

5. Tee tarkistusnappula

.Name ***Tarkista***

6. Koodaa nappulan tapahtuma-aliohjelma, joka käy kaikki nappulat järjestyksessä läpi ja ilmoittaa, jos rivin päässä oleva valinta on valittuna.

```
Private Sub Tarkista_Click()
Dim I
For I = 0 To 29
  If Valinta(I).Value=True And (I Mod 5 = 0 Or I Mod 5 = 4) Then
     MsgBox (“Rivi “ & I \ 5 + 1 & “ sarake “ & Chr(I Mod 5+65))
   End If
 Next
```

11. LISÄÄ KOMPONENTTEJA

Tässä luvussa otetaan käyttöön uusi komponentti MSFlexGrid, joka ei kuulu vakio-komponentteihin. Lisäksi tutustutaan ajastimen käyttöön kahdessa erilaisessa sovelluksessa.

Esimerkki9:VALTION TULOVERO

Ohjelmassa lasketaan palkansaajan valtiolle maksama vuotuinen vero, joka perustuu taulukoituihin tuloluokkiin.

- ***useamman lomakkeen sovellus***
- ***ohjelmamoduuli***
- ***MsFlexGrid***

Lomake 1

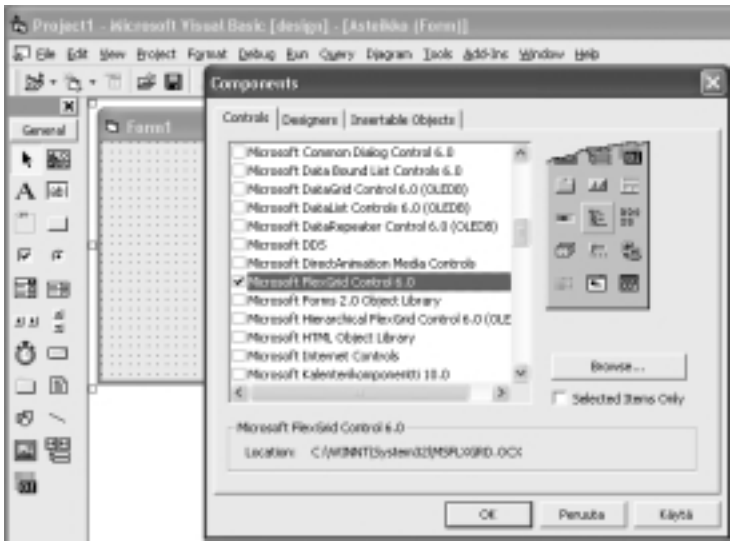
0. Aloita uusi sovellus (Standard.EXE)

1. Nimeä lomake ja valitse sovelluksellesi ikoni, joka näkyy sekä ohjelmapalkissa että kuvakkeena Windowsin resurssienhallinnassa.

.Name	***Asteikko***
.Caption	***VALTION TULOVEROASTEIKKO***
.Icon	***valitse ICO-tyyppinen bittikartta selaamalla***

2. Tee lomakkeelle MSFlexGrid, jonka nimeät Gridiksi

.Name	***Grid***
.Fixed Cols	***0***
.Cols	***3***
.Rows	***6***



Lisää komponentteja saa painamalla työkalupaletista hiiren oikealla näppäimellä.

3. Alusta Grid seuraavilla lauseilla Form Load –tapahtumassa. Muista, että Asteikko.Show ei takaa tietojen siirtymistä Gridiin, vaan lomakkeen tietojen päivittäminen tapahtuu sitä kutsuttaessa

```
Private Sub Form_Load()
  Asteikko.Refresh.
  Grid.TextMatrix(0,0)=”Verotettava tulo”
  Grid.TextMatrix(0,1)=”Vero alarajan kohdalla”
  Grid.TextMatrix(0,2)=”Vero alarajan ylittävästä osasta”
  Grid.TextMatrix(1,0)=”11600-14400”      ’rivi 1
  Grid.TextMatrix(1,1)=”8”
  Grid.TextMatrix(1,2)=”12.5”
  Grid.TextMatrix(2,0)=”14400-20000”      ’rivi 2
  Grid.TextMatrix(2,1)=”358”
  Grid.TextMatrix(2,2)=”16.5”
  Grid.TextMatrix(3,0)=”20000-31200”      ’rivi 3
  Grid.TextMatrix(3,1)=”1282”
  Grid.TextMatrix(3,2)=”22.5”
  Grid.TextMatrix(4,0)=”31200-55200”      ’rivi4
  Grid.TextMatrix(4,1)=”3802”
  Grid.TextMatrix(4,2)=”28.5”
  Grid.TextMatrix(5,0)=”55200-”           ’rivi 5
  Grid.TextMatrix(5,1)=”10642”
  Grid.TextMatrix(5,2)=”35.5”
  Grid.ColWidth(0) = 1220
  Grid.ColWidth(1) = 1680
  Grid.ColWidth(2) = 2300
End Sub
```

4. Tallenna projektisi tässä vaiheessa. Anna lomakkeelle nimeksi asteikko ja projektille nimi tulovero. Näin muodostuu tiedostot

ASTEIKKO.FRM
TULOVERO.VBP

Tallenna projekti myös ajettavaksi ohjelmaksi. File Make Tulovero.EXE.

Lomake 2

1. Lisää toinen lomake sovellukseesi (Project | Add Form | Form) ja nimeä se

.Name	*Kysely*
.Caption	*Anna tulotietosi*

2. Määrittele projektin ominaisuuksista Kysely-lomake aktiiviseksi sovellusta käynnistettäessä

Project | Veroprojekti Properties | Startup Project | Kysely

3. Tee lomakkeelle sekä teksti-kenttä että sen opaste

.Caption	*Tulot ennen veroja*
.Name	*Tulot*
.Text	*(tyhjä)*
.Alignment	*Right Justify*

4. Tee lomakkeelle toinenkin tekstikenttä ja sen opaste ja anna sille nimeksi Verot ja anna alkuarvo

.Caption	*Verot markoissa*
.Name	*Verot*
.Text	*0*
.Alignment	*Right Justify*

5. Tee komentopainike

.Name	*Laske*
.Caption	*Laske*

6. Kirjoita Laske-nappulaan Click-tapahtumakäsittelijä

```
Private Sub Laske_Click()
  Asteikko.Show         'lomake näkyviin
  Asteikko.Refresh      'päivitä tiedot
  Asteikko.Left = 550   'siirrä lomaketta oikealle
  Asteikko.Top = 400    'siirrä lomaketta alas
End Sub
```

7. Pienennä Kysely-lomake juuri sopivan kokoiseksi

8. Siirrä Kysely-lomaketta vasemmalle

```
Private Sub Form_Load()
  Kysely.Left=8000
End Sub
```

9.Tallenna sovelluksesi. Huomaa, että toisella kerralla tallennettaessa kaikki tallentuu File Save Project –komennolla. Visual Basic kysyy Kysely-lomakkeen tallentamista, jonhon vastaat Kyllä.

Moduli Laske

1. Tee projektiisi uusi moduli laskutoimituksia varten (Project | Add module | Open)

2. Nimeä moduuli

.Name	*Laskut*

3. Kirjoita moduuliin funktio, jolla lasketaan kokonaisvero valtion verotuksessa kaavalla

*Algoritmi: VVero=VeroTaulukosta+(Tulot-TulonAlarajaTaulukosta)*ProsenttiTaulukosta/100*

```
Function VVero (parvero,partulot,paralaraja,parprosentti)
  VVero=parvero+(partulot-paralaraja)*parprosentti/100
End Function
```

4. Koska tapahtumat käynnistyvät Kysely-lomakkeesta, kirjoitetaan kaikki muu koodi funktiota lukuunottamatta Laske-nappulaan. Ennen taulukon läpikäyntiä katsotaan, onko tulot taulukkoarvoa pienemmät tai sitten suurimmassa tuloluokassa.

Otetaan Verot.Text-kenttään arvo taulukosta toistorakenteessa. Koska tulon alaraja löytyy helpoiten yhtä riviä alempaa, tulee vero alarajan kohdalla, tulot alarajan kohdalla ja prosentti etsiä riviltä toistolaskuri-1. Tulot on kokoajan tallessa Tulot.Text-kentässä. Verot.Text joka sisälsi aiemmin veron alarajan kohdalla, muuttuu funktiokutsun jälkeen osoittamaan kokonaisveroa.

Toistorakenteessa aloitetaan arvosta 14400, koska testataan sitä pienempiä arvoja. Jos oikea arvoalue löytyy, merkitään totuusarvomuuttujaan FOUND (löytyi), eikä sen jälkeen uusia arvoja tarvitse etsiä. Samalla toistokerralla sijoitetaan globaaleihin muuttujiin muut tarvittavat arvot (TulotTaulukosta, ProsenttiTaulukosta)

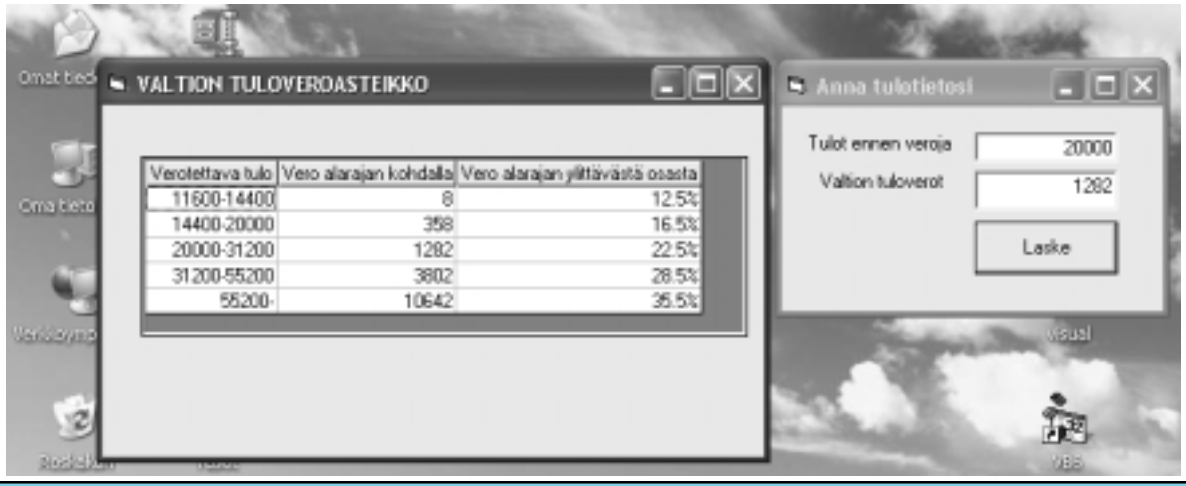
Koska sovellusta on helpompi käyttää Enterin painalluksella käynnistetään Laske-nappulan aliohjelma myös siitä.

Seuraavassa Kysely-lomakkeen koodi kokonaisuudessaan

```
Private Sub Form_Load()
  Kysely.Left = 8000
End Sub

Private Sub Laske_Click()
Dim Found, Y, ProsenttiTaulukosta, TulotTaulukosta

  Asteikko.Show
  Asteikko.Refresh
  Asteikko.Left = 550
  Asteikko.Top = 400
  If Val(Tulot.Text) < 11600 Then
    Found = True
    Verot.Text = 0
    TulotTaulukosta = 0
    ProsenttiTaulukosta = 0
  End If
  If Val(Tulot.Text) > 55200 Then
    Found = True
    Verot.Text = 10642
    TulotTaulukosta = 55200
    ProsenttiTaulukosta = 35.5
  End If
For Y = 2 To 5
  If Not (Found) And Val(Tulot.Text) <= &_
    Left(Asteikko.Grid.TextMatrix(Y, 0), 5) Then
   Found = True
   Verot.Text = Asteikko.Grid.TextMatrix(Y - 1, 1)
   TulotTaulukosta = Left(Asteikko.Grid.TextMatrix(Y - 1, 0), 5)
   ProsenttiTaulukosta = Val(Asteikko.Grid.TextMatrix(Y - 1, 2))
  End If
 Next Y
Verot.Text = VVero(Val(Verot.Text), Val(Tulot.Text), TulotTaulukosta,
ProsenttiTaulukosta)
End Sub

Private Sub Tulot_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
    Laske_Click
  End If
End Sub

Private Sub Form_UnLoad(Cancel As Integer)
  If MsgBox("Haluatko lopettaa?",vbYesNo)=vbYes Then
    Cancel=0
    UnLoad Asteikko
  Else
    Cancel=1
  End If
End Sub
```

Huomioita ohjelmasta

MSFlexGrid-komponentin indekseistä ensimmäinen osoittaa riviä ja toinen saraketta. Molemmat indeksit alkavat otsikkoriveistä/-sarakkeista riippumatta nollasta. Ohjelmasta poistuminen pitäisi tapahtua ainoastaan, kun ohjelmoija sen haluaa tapahtuvan. Käytännössä poistumislupaa voidaan testata Form_UnLoad-aliohjelmassa. Cancel-parametrin tulee olla nolla, kun sovelluksen lopettaminen sopii. Toisaalta Asteikko-lomake suljetaan UnLoad-komennolla.



Valtion tuloveron laskemisessa käytetään esimerkin kaltaisesti taulukoituja tuloluokkia.

Esimerkki11:Liikennevalot

Tee liikennevalot tekstikehysten taustaväreinä siten, että ne vaihtuvat punaisesta vihreään päin punainen ja oranssi palaen yht'aikaisesti.

- ***ajatus paperille ja paperilta ohjelmaksi***
- ***totuusarvo- eli booleanmuuttujat***
- ***ajastintoiminnot***

1. Ajatus kannattaa panna aluksi paperille (mielikuvana siitä mitä todellisuudessa tapahtuu)

Ennen ajastinluupia on asetettava alas-muuttuja tosiarvoon ja laskuri nollaksi

jos mennään alaspäin niin kasvatetaan laskuria
muuten vähennetään laskuria

Ajastinlaskuri alaspäin mentäessä	***Ajastinlaskuri ylöspäin tultaessa***
1 punainen päälle	***1 alas muuttuu todeksi***
2	***2***
3	***3***
4	***4***
5 oranssi päälle	***5 punainen päälle (oranssi pois)***
6	***6***
7 vihreä päälle(punainen ja oranssi pois)	***7 oranssi päälle(vihreä pois)***
8	***8***
9	***9***
10 alas muuttuu epätodeksi 10	

2. Mielikuvasta VB-ohjelmaksi. Tee sovellukseen kolme tekstilaatikkoa siten, että kopioitaessa niistä muodostuu kontrollirakennetaulukko.

.Name	***Valo(0)***
	Valo(1)
	Valo(3)

3. Määrittele globaalit muuttujat ja alusta ne Load-tapahtumassa

```
Option Explicit
Dim Y,Alas

Private Sub Form_Load()
  Y = 0
  Alas = True
End Sub
```

4. Tee lomakkeelle ajastin ja kirjoita sen koodi

.Name	*Ajastin*
.Interval	*500*

```
Private Sub Ajastin_Timer()
If Alas Then
  Y = Y + 1
  If Y = 1 Then
    valo(0).BackColor = vbRed
  ElseIf Y = 5 Then
    valo(1).BackColor = RGB(155, 44, 44)
  ElseIf Y = 7 Then
    valo(0).BackColor = vbBlack
    valo(1).BackColor = vbBlack
    valo(2).BackColor = vbGreen
  ElseIf Y = 10 Then
    Alas = False
  End If
Else
  Y = Y - 1
  If Y = 7 Then
    valo(2).BackColor = vbBlack
    valo(1).BackColor = RGB(155, 44, 44)

  ElseIf Y = 5 Then
    valo(1).BackColor = vbBlack
    valo(0).BackColor = vbRed
  ElseIf Y = 1 Then
    Alas = True
  End If
End If
End Sub
```

Huomioita ohjelmasta

Jos on aikaisemmin ohjelmoinut DOS-ohjelmia, tulee helposti mieleen käyttää jotakin muuta tapaa ratkaista ongelma - esimerkiksi Do .. Until -toistorakenteen avulla. Pitää kuitenkin muistaa, että kaikki muut kuin ajastimeen sidotut toistot tapahtuvat, kun on niiden suoritusvuoro ohjelmakoodissa, eikä prosessori jää odottamaan itserakennettuja viiveitä, vaan siirtyy suoraan seuraavaan ohjelmakäskyyn.

Esimerkki12: Tekstiviesti

Ohjelmassa matkitaan tekstiviestilähettämistä matkapuhelimella

- ***MouseDown- ja MouseUp-näppäintapahtumat***
- ***ASCII-koodi***

0. Sovelluksessa on useita samankaltaisia kontrollirakenteita - sekä ominaisuuksiltaan että tapahtumiltaan. Tällainen sovellus voidaan ratkaista ainoastaan miettimällä, kuinka kopioidun kontrollirakennetaulukon indeksiä voidaan hyödyntää kunkin komponentin ominaisuuksissa ja niissä tapahtumissa, mitkä rakenne käynnistää.

1. Mietitään mitä globaaleja muuttujia ohjelmassa voitaisiin tarvita

- ***laskuri (osoittaa mones merkki nappulassa on menossa)***
- ***indeksi (osoittaa mones nappula on kysymyksessä)***
- ***alussa (muuttuu epätodeksi, kun ensimmäinen merkki nappulasta on lisätty viestiin)***

2. Mitä tapahtumia?

Form_Load()

- ***ajastin ei saa olla päällä vielä tässä kohdassa***
- ***painikkeet saavat merkkinsä (toistetaan 9 kertaa)***

```
For X=0 to 8
  nappula(X).Caption = Chr(X*3 + 65) + Chr(X*3 + 66)+Chr(X*3+67)
Next X
```

Toistolaskurin alkuarvo kannattaa valita silloin nollaksi, kun laskurilla kerrotaan jotakin lukua - tällöin ensimmäisellä toistokerralla kertolaskun arvo ei muutu

Mouse_Down() ja Mouse_Up()

- ***nappula alaspainettu: otetaan indeksi talteen***
- ***laskuri, joka osoittaa ajankulumista =>0***
- ***ensimmäinen merkki todeksi***
- ***ajastin aktivoidaan - Mouse_Up-aliohjelmassa pysäytetään***

Timer_Timer()

- ***kasvatetaan laskuri kunnes kulunut kolme aikayksikköä, jolloin nollataan laskuri***
- ***tekstikenttään sopiva merkki***
- ***kasvata ensimmäisellä kerralla tekstikenttää +chr(indeksi+65+laskuri)***
- ***muilla toistokerroilla lisää merkki jonon viimeisen merkin tilalle***

3. Lopulta suunnitelman pohjalta VB-ohjelma pitäisi syntyä helposti. Lomakkeella tulisi olla monirivinen tekstikenttä viesti ja yhdeksän kontrollirakennetaulukoksi kopioitua painiketta nimiltään nappula(0)...nappula(8), sekä ajastin.

```
Option Explicit
Dim laskuri, Alussa, Indeksi

Private Sub ajastin_Timer()
If Alussa Then
   viesti.Text = viesti.Text + Chr(Indeksi * 3 + 65 + laskuri)
   Alussa = False
Else
   viesti.Text = Left(viesti.Text, Len(viesti.Text) - 1) + Chr(Indeksi * 3 + 65 + laskuri)
End If
If laskuri < 2 Then
   laskuri = laskuri + 1
Else
   laskuri = 0
End If
End Sub

Private Sub Form_Load()
Dim X
  For X = 0 To 8
     nappula(X).Caption=Chr(X * 3 + 65)+Chr(X * 3 + 66)+Chr(X * 3 +67)
  Next X
   ajastin.Enabled = False
End Sub

Private Sub nappula_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
laskuri = 0
First = True
ajastin.Enabled = True
Indeksi = Index
End Sub

Private Sub nappula_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
   ajastin.Enabled = False
End Sub
```

Huomioita ohjelmasta

Esimerkissä on jouduttu julkaisuteknisistä syistä pienentämään kirjasinkokoa muutamassa kohdassa. VB:ssä harvoin rivi loppuu kesken, mutta joskus selkeyden takia koodia joudutaan jatkamaan seuraavalta riviltä kesken komennon. Tällöin jatkomerkkinä käytetään rivin lopussa merkkejä &_.

Tässä ohjelmassa tarvitaan ASCII-merkistön tuntemusta. On muistettava, että merkit ovat järjestyksessä ainoastaan englanninkielisen aakkosmerkistön (A..Z) merkkien kesken. Seuraavassa ASCII-desimaalit omissa (yksinkertaistetuissa) ryhmissänsä esimerkkeineen.

ryhmä	esimerkki VB:ssä	ASCII-merkki
0) kontrollimerkit	CHR(7)	^G =BELL(ääni)
1)erikoismerkit	CHR(32)	SPACE(välilyönti)
2)numerot	CHR(48)..CHR(57)	0..9
3)isot aakkoset	CHR(65)..CHR(90)	A..Z
4)pienet aakkoset	CHR(97)..CHR(122)	a..z

==========<=7 bittiä====================8 bittiä>====================

5)graafiset merkit, joiden joukossa eri kielialueiden omat merkit (ÄÖÅ jne)

12. VIRHEKÄSITTELYSTÄ

Ohjelman käyttömukavuuden ja toiminnallisuuden kannalta on tärkeää käyttää ns. tuotantoversioissa mahdollisimman paljon aikaa virhekäsittelyyn. Ajonaikaisia virheitä aiheuttavat mm. tiedosto-operaatiot, jossa oletetaan tiedoston löytyvän tietystä paikasta ja käyttäjän syöttämät numerot. Ilman virhekäsittelyä ohjelman käyttö hidastuu tai loppuu kokonaan.

Seuraavassa esimerkissä syntyy virhetilanne, jos toiseen tai kumpaankin tekstikenttään syötetään merkkitietoa, jätetään tyhjäksi, käytetään väärää desimaalierotinta (virhe 13) tai käytetyt numerot ovat liian suuria (virhe 6).

```
Private Sub Command1_Click()
Dim luku%
On Error GoTo Virhe
  luku% = Text1.Text * Text2.Text
MsgBox ("…jatkuu")
Exit Sub
Virhe:
  MsgBox (Err.Number)
Resume Next
End Sub
```

Jos yo. ohjelmassa ei haluta jatkaa aliohjelman suoritusta virheen jälkeen, jätetään esimerkin MsgBox("..jatkuu") koodista pois ja korvataan Resume Next –rivi Exit Sub –komennolla.

Tutustu internetistä lisää VisualBasicin virhekäsittelyyn. Hakusanoina kannattaa käyttää

visual basic tutorial

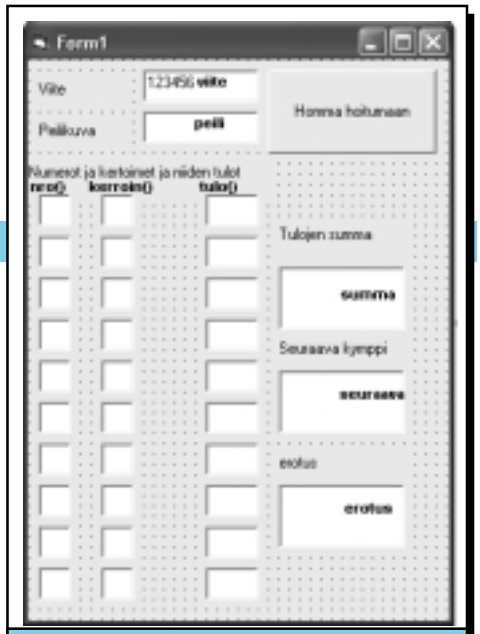
ja lisäksi error handling.

Esimerkki 13: Viitenumero

Ohjelmassa lisätään viitenumeron loppuun oikeansuuruinen tarkistusnumero. Viitenumeroa tulkitaan oikealta päin kertoen kukin vuorossa oleva luku kertoimella, joka vaihtuu järjestyksessä 7-3-1. Kun kaikkien lukuparien tulot on laskettu yhteen, etsitään seuraava täysi kymmenluku, josta yo. summa vähennetään.

- ***tiedon tarkistaminen laskemalla (MD5)***
- ***ongelman algoritmin ratkaiseminen paloina***
- ***StrReverse-funktio***

1. Tee kuvan mukainen lomake. Kopioi tekstilaatikot (Nro, Kerroin, Tulo) kontrollirakennetaulukoksi.
2. Muuta viitenumero pelikuvaksi (StrReverse)
3. Sijoita luvut yksi kerrallaan alekkaisiin tekstilaatikoihin (mid)
4. Kirjoita lukujen viereen kertoimet (7-3-1)
5. Laske rivien tulot
6. Laske alekkaisten rivien summa
7. Vähennä summa seuraavasta tasasta kymmenluvusta



Vaikea ongelma aukeaa, kun purkaa sen pieniin osiin.

Jos käytä VB5:sta, joudut tekemään StrReverse-funktion itse seuraavalla tavalla:

```
Function StrReverse(jono)
Dim X,Apujono
  For X=0 to Len(jono)-1
    Apujono=apujono& Mid(jono,len(jono)-x,1)
  Next X
  StrReverse=Apujono
End Function
```

Muuten ohjelman ratkaisu etenee ohjelman ainoasta komentopainikkeesta näin

```
Private Sub Command1_Click()
Dim Y
Dim summm As Integer
Dim k(3)
k(0) = „7“
k(1) = „3“
k(2) = „1“
summa = 0
peili.Text = StrReverse(viite.Text)
For Y = 0 To Len(peili.Text) - 1
  Nro(Y).Text = Mid(peili.Text, Y + 1, 1)
  kerroin(Y).Text = k(Y Mod 3)
  tulo(Y).Text = Nro(Y).Text * kerroin(Y).Text
  summm = summm + Val(tulo(Y).Text)
Next Y
summa.Text = summm
seuraava.Text=(Mid(summa.Text, Len(summa.Text) - 1, 1) + 1) * 10
erotus.text= seuraava.text-summa.text
viite.Text = viite.Text & erotus.text
End Sub
```

Huomioita ohjelmasta

Edellinen oli vain esimerkki ongelman pilkkomisesta. Todellisuudessa kaikki voidaan kirjoittaa lyhyenä funktiona, johon parametrina viedään viitenumeron alku ja joka palauttaa MD5-tarkistetun viitenumeron.

Esimerkki 14: Valuutanvaihto

Kurssi ilmoitetaan sen suhteessa euroon siten, kuinka monta kertaa kyseistä valuutta tarvitaan yhden euron hankintaan

Maa	valuutta	lyhenne	kurssi
Yhdysvallat	dollari	USD	1,1748
Englanti	punta	GBP	0,7047
Ruotsi	kruunu	SEK	9,1060
Latvia	lati	LVL	0,6578
Viro	kruunu	EEK	15,6466

Jos listaobjektin tietoja halutaan käyttää sen indeksin avulla, näyttää ohjelmalause esim. tällaiselta

```
List1.List(indeksi)
```

tai, jos listan riviä on painettu hiirellä

```
List1.List(List1.ListIndex)
```

1. Harjoitellaan listaobjektin käyttöä, koska valtion tuloveroa laskettaessa käytettiin MSFlexGridiä. Tee ohjelmaasi neljä ListBoxia yo. taulukon mukaisesti

2. Valikkoon tulee seuraavat valinnat

Syötä kurssit	*Valuuttakurssit luetaan toistorakenteessa.*
Uusi valuutta	*Kaikkiin listoihin lisätään uuden valuutan tiedot, jotka kysytään InputBoxilla*
Valuutanvaihto	*Kysy vaihdettavan valuutan lyhenne, valuutan määrä ja se valuutta mihin vaihdetaan ja laske uuden valuutan määrä.*

```
'Aliohjelmassa luetaan kaikkien listassa olevien valuuttojen uudet kurssit
'vanha kurssi on oletusarvona
Private Sub kurssit_Click()
Dim I
'käydään toistossa läpi kaikki listan alkiot
For I = 0 To Kurssi.ListCount - 1
  Kurssi.List(I) = InputBox("Anna kurssi", Lyhenne.List(I), Kurssi.List(I))
Next I
End Sub

'Aliohjelmassa lisätään kaikkiin listoihin uusi rivi uusilla tiedoilla
Private Sub uusi_Click()
Dim apu
  apu = InputBox("Anna uusi maa")
  Maa.AddItem (apu)
  apu = InputBox("Anna uusi valuutta")
  Valuutta.AddItem (apu)
  apu = InputBox("Anna uusi lyhenne")
  Lyhenne.AddItem (apu)
  apu = InputBox("Anna uusi kurssi")
  Kurssi.AddItem (apu)
End Sub

'Aliohjelmassa kysytään sekä lähtö- että kohdevaluutta ja valuutan määrä
'ja lasketaan uuden valuutan määrä
Private Sub vaihto_Click()
Dim LValuutta, KValuutta, Maara, L, K, LKurssi, KKurssi, I
LValuutta = InputBox("Anna lähtövaluutta")
KValuutta = InputBox("Anna kohdevaluutta")
Maara = InputBox("Anna lähtövaluutan määrä")
```

```
'Lähtö- ja kohdeindeksit alustetaan sellaisella arvolla, jota ei löytämisen
jälkeen voi olla
L = -1
K = -1
'Toistossa käydään kaikki lyhennerivit läpi
For I = 0 To Lyhenne.ListCount – 1
'jos lyhennerivin sisältö vastaa hakuavainta …
  If Lyhenne.List(I) = LValuutta Then
    L = I
  End If
  If Lyhenne.List(I) = KValuutta Then
    K = I
  End If
Next I
'jos jompaakumpaa lyhennettä ei löytynyt, tulostetaan viesti…
If L = -1 Or K = -1 Then
  MsgBox ("Ainakin toinen valuutoista puuttuu listalta!")
…muuten lasketaan ja tulostetaan valuutanvaihtotapahtuma
Else
'Lähtövaluutan määrä*kohdevaluutan kurssi/lähtövaluutan kurssi
  MsgBox (Val(Maara) * Val(Kurssi.List(K)) / Val(Kurssi.List(L)))
End If
```

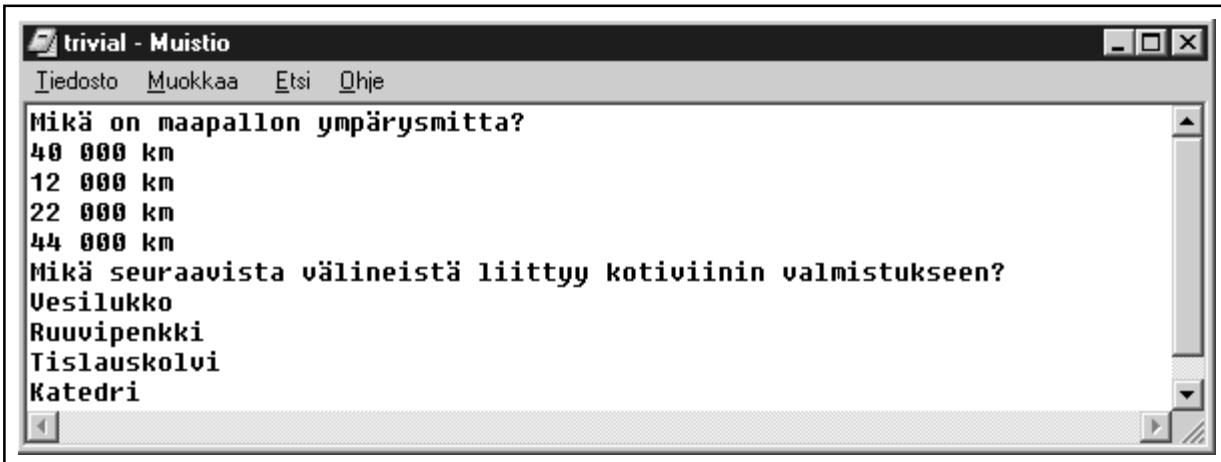
End Sub

Esimerkki 15: Trivial Pursuit

Tee sovellus, jossa aluksi siirrät tekstitiedostoon kirjoitetut kysymykset lomakkeen näkymättömään ListBoxiin. Sovelluksessa ao. kysymys tulostetaan Label-kenttään ja vastaukset satunnaisessa järjestyksessä Option-valikoksi. Lisäksi sovelluksessa näkyy koko ajan oikeiden vastausten suhde kaikkiin vastauksiin ja edellisen kysymyksen oikea vastaus.

```
Dim Ed, oikeat, yht, oikea, arvaus, oikein
Private Sub Command1_Click()
Dim i
Dim indeksi
Randomize
Edellinen.Caption = "Edellisen tehtävän oikea ratkaisu oli " & Ed
For i = 0 To 2
  Option1(i).Value = False
Next i
yht = yht + 1
indeksi = Int(Rnd * (List1.ListCount / 5)) * 5
Kysymys.Caption = List1.List(indeksi)
oikea = Int(Rnd * 4)
Option1(oikea).Caption = List1.List(indeksi + 1)
Ed = List1.List(indeksi + 1)
paikka = 0
For i = 0 To 2
  If oikea = i Then
    paikka = 1
  End If
  Option1(paikka + i).Caption = List1.List(indeksi + i + 2)
Next i
If oikein Then
  oikeat = oikeat + 1
  oikein = False
End If
Tilanne.Caption = oikeat & "/" & yht - 1
End Sub
```

```
Private Sub Form_Load()
Dim rivi
oikeat = 0
yht = 0
Open ”C:\TEMP\TRIVIAL.TXT” For Input As #1
Do Until EOF(1)
  Line Input #1, rivi
  List1.AddItem (rivi)
Loop
Command1_Click
End Sub

Private Sub Option1_Click(Index As Integer)
  arvaus = Index
  MsgBox (arvaus & ”=” & oikea & (oikea = arvaus))
  If arvaus = oikea Then oikein = True
  Command1_Click
End Sub
```



Trivial Pursuit -pelissä kysymykset on aluksi kirjoitettu tekstitiedostoon siten, että kysymyksen alla on aina oikea vastaus ja sen jälkeen kolme väärää vastausta. Ohjelma sekoittaa joka kerralla vaihtoehdot satunnaisjärjestykseen.

12. HARJOITUKSIA

Esimerkki 16: Kilometrikorvaus

Tee sovellus, jossa on valmiina taulukoituna kauppamatkustajan tavanomaisimpia reittejä (lähtökunta-loppukunta-matka km). Sovelluksessa sijoitetaan taulukkoon ajankohta ja reittitiedot ja voidaan valita kuukausi ja sovellus laskee matkojen kokonaispituuden sekä kilometrikorvaukset (38 senttiä/km).

Aloita sovelluksen teko siten, että Form1_Load ()-aliohjelmassa siirrät riittävän määrän matkatietoja kilometritaulukkoon. Kilometritaulukossa jotain riviä painamalla sijoitetaan päivämäärä ja painetun rivin indeksi ajopäiväkirjaan, jos pvm-kenttä ei ole tyhjä.

Jatka nappulasta "Laske korvaukset" käymällä pvm-kentän osoittaman kuukausinumeron perusteella kaikki pkirja-taulukon rivit läpi ja poimi kuukautta vastaavat rivi-indeksit ja laske kilometrit-taulukon vastaavan matkan ja kilometrikorvauksen tulojen summa. Jos pvm-kentässä ei ole kuukautta, lasketaan koko pkirjataulukon korvaukset.

Ja poista pkirja-taulukon rivejä, jos vastaus kysymykseen "Poistetaanko valittu rivi?" vastataan myöntävästi.

Lisätehtävänä tee sovellukseen alasvetovalikot, joissa on valittavana päivä (1-31) ja kuukausi (1-12). Tee toiminto, jolla haet pkirja-taulukosta päiväyksiä vastaavat matkat.

Käytä pkirja-taulukon päiväyksen pilkkomiseen month-funktiota. Lisäksi otetaan käyttöön uusi Windowsin datatyyppi ns. MSFlexGrid, jota kannattaa suomen kielellä kutsua ehkä taulukoksi ja nimetä ohjelmassa tauluksi.

Taulukon käsittelyssä voidaan käyttää seuraavia tapoja:

Otsikkorivit ja sarakkeet ilmoitetaan lukumäärinä

```
Taulu.FixedRows=1
Taulu.FixedCols=0
```

Rivien ja sarakkeiden lisääminen taulukkoon

```
Taulu.Rows = Taulu.Rows + 1
Taulu.Cols = Taulu.Cols + 1
```

Solun sisällön muuttaminen ja tulostaminen

```
Taulu.TextMatrix(1, 1) = "Aku"
MsgBox(Taulu.TextMatrix(1,1))
```

Hiirellä taulukosta painettu kohta löytyy seuraavilla metodeilla

```
Taulu.MouseCol
Taulu.MouseRow
```

Tässä tulostetaan hiirellä painetun solun sisältö

```
MsgBox(taulu.TextMatrix(taulu.MouseRow,taulu.MouseCol))
```

```
Private Sub Command1_Click()
Dim i
   For i = 1 To pkirja.Rows - 1
  If rpvm.Text <> “päivä” Then
    If rkk <> “kk” Then
      If Month(pkirja.TextMatrix(i, 0)) = Val(rkk.Text) And
(Day(pkirja.TextMatrix(i, 0)) = Val(rpvm.Text)) Then
        MsgBox (kilometrit.TextMatrix(pkirja.TextMatrix(i, 1), 2))
      End If
    Else
      If (Day(pkirja.TextMatrix(i, 0)) = Val(rpvm.Text)) Then
        MsgBox (kilometrit.TextMatrix(pkirja.TextMatrix(i, 1), 2))
      End If
    End If
  Else
    If Month(pkirja.TextMatrix(i, 0)) = Val(rkk.Text) Then
        MsgBox (kilometrit.TextMatrix(pkirja.TextMatrix(i, 1), 2))
    End If
  End If
Next i
rpvm.Text = “päivä”
rkk.Text = “kk”
End Sub

Private Sub Form_Load()
Dim i
  For i = 1 To 31
    rpvm.AddItem (i)
    If i < 13 Then rkk.AddItem (i)
  Next i
  pvm.Text = Date
  kilometrit.Rows = 5
  kilometrit.ColWidth(0) = 500
  kilometrit.ColWidth(1) = 1500
  kilometrit.ColWidth(2) = 1500
  kilometrit.ColWidth(3) = 600
  kilometrit.TextMatrix(0, 1) = “Alku”
  kilometrit.TextMatrix(0, 2) = “Loppu”
  kilometrit.TextMatrix(0, 3) = “km”
  kilometrit.TextMatrix(1, 0) = „1“
  kilometrit.TextMatrix(1, 1) = „Riihimäki“
  kilometrit.TextMatrix(1, 2) = „Lahti“
  kilometrit.TextMatrix(1, 3) = „55“
  kilometrit.TextMatrix(2, 0) = „2“
  kilometrit.TextMatrix(2, 1) = „Riihimäki“
  kilometrit.TextMatrix(2, 2) = „Kouvola“
  kilometrit.TextMatrix(2, 3) = „120“
  kilometrit.TextMatrix(3, 0) = „3“
  kilometrit.TextMatrix(3, 1) = „Riihimäki“
  kilometrit.TextMatrix(3, 2) = „Helsinki“
  kilometrit.TextMatrix(3, 3) = „50“
  kilometrit.TextMatrix(4, 0) = „4“
  kilometrit.TextMatrix(4, 1) = „Riihimäki“
  kilometrit.TextMatrix(4, 2) = „Oulu“
  kilometrit.TextMatrix(4, 3) = „550“
  pkirja.TextMatrix(0, 0) = „Pvm“
  pkirja.TextMatrix(0, 1) = „Matka“
  pkirja.ColWidth(1) = 600
End Sub
```

```
Private Sub MSFlexGrid1_Click()

End Sub

Private Sub kilometrit_Click()
  If pvm.Text <> "" Then
    pkirja.Rows = pkirja.Rows + 1
    pkirja.TextMatrix(pkirja.Rows - 1, 0) = pvm.Text
    pkirja.TextMatrix(pkirja.Rows - 1, 1) = kilometrit.MouseRow
  End If
End Sub

Private Sub Laske_Click()
Dim i, pituus
pituus = 0
For i = 1 To pkirja.Rows - 1
  If Month(pkirja.TextMatrix(i, 0)) = Val(pvm.Text) Or pvm.Text = "" Then
    pituus = pituus + Val(kilometrit.TextMatrix(pkirja.TextMatrix(i, 1), 3))
  End If
Next i
korvaukset.Text = pituus * 0.38 & " •"
End Sub

Private Sub pkirja_Click()
'poistettava rivi on tyypiltään pitkä kokonaisluku
Dim privi&

privi&=pkirja.MouseRow
  If MsgBox("Haluatko todella poistaa tämän rivin?", vbYesNo) = vbYes Then
    pkirja.RemoveItem (pkirja.privi&)
  End If
End Sub
```

Esimerkki 17: Drag&Drop

Tee sovellus, jossa viet "VEDÄ JÄ PUDOTA" -tyylin palautuspulloja automaattiin. Kun tuote vedetään automaattiin, verrataan sen indeksi-kenttää siten, että jos numero on 0 kokonaispalautussumma kasvaa 10 sentillä, 1 summa kasvaa 15 sentillä ja tagin ollessa 2 kasvatetaan summaa 40 sentillä. Samalla lisää indeksin osoittamaa pullot-taulukon alkiota yhdellä, jotta sovelluksen lopuksi voidaan eri pullomäärät tulostaa näkyviin.

Edellytyksenä Drag&Drop-toiminnoille on objektien DragIcon ja DragMode -määrittelyjen muuttaminen oletusarvoista. Itse DragDrop-tapahtuma määritellään siihen objektiin, johon objektit pudotetaan.

Indeksin saat käyttöön kopioimalla kontrollirakenteita taulukoksi. Riittää , kun vastaat objektia kopioitaessa kysymykseen "Do you want to create a Control Array?" myönteisesti.

```
Dim pullot(3)
Dim yht

Private Sub Image1_DragDrop(Source As Control, X As Single, Y As Single)
    Source.Visible = True
    If Source.Index = 0 Then
        yht = yht + 0.1
    ElseIf Source.Index = 1 Then
        yht = yht + 0.15
    Else
      yht = yht + 0.4
    End If
    Summa.Text = yht
    pullot(Source.Index) = pullot(Source.Index) + 1
    MsgBox (pullot(0) & “, “ & pullot(1) & “, “ & pullot(2))
End Sub
```

13. TIETOKANTATOIMINNOT

Tässä luvussa tutustutaan ACCESS-tietokantojen käyttöön VB-ohjelmissa. Tee tietokanta ACCESSin 7.0-formaatissa joko ACCESSissa tai VB:n Visual DataManagerissa.

Data-kontrollirakenteessa on "sisäänrakennetut" navigointinäppäimet. Datakomponentista pitää määritellä tietokannan tiedostonimi (database) ja lomakkeelle halutun taulukon nimi (RecordSet).

Tietokannan kenttien sisällöt voidaan siirtää haluttuihin oikenmuotoisiin Windows-objekteihin määrittämällä näille DataSource (aiemmin tehty datakomponentti) ja objektiin sopiva kenttä (Datafield).

Data-komponentin käyttöä voi monipuolistaa piilottamalla sen ja tekemällä omat kontrollirakenteet tietokannan selaamiseksi ja päivittämiseksi. RecordSetin käsittelyssä voidaan käyttää mm. seuraavia komentoja:

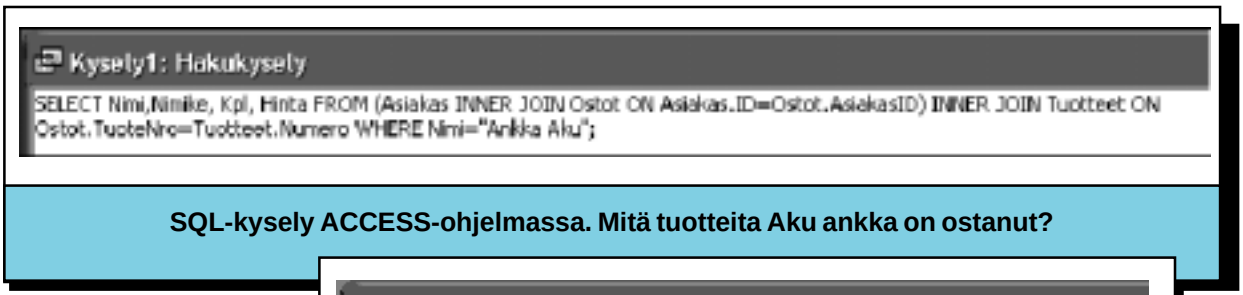
```
Data1.RecordSet.AddNew          'lisää tyhjän tietueen tietokannan loppuun
Data1.RecordSet.Delete          'poistaa kohdalla olevan tietueen
Data1.RecordSet.Update          'päivittää muutokset tietokantaan
Data1.RecordSet.Seek            'etsii hakuavainta vastaavan tietueen kohdalle

Data1.RecordSet.MoveFirst       'siirtyminen haetun tietorakennelman alkuun
Data1.RecordSet.MoveNext        'siirtyminen seuraavaan tietueeseen
Data1.RecordSet.MovePrevious    'siirtyminen edelliseen tietueeseen
Data1.RecordSet.MoveLast        'siirtyminen tiedoston loppuun
```

SQL ja Visual Basic

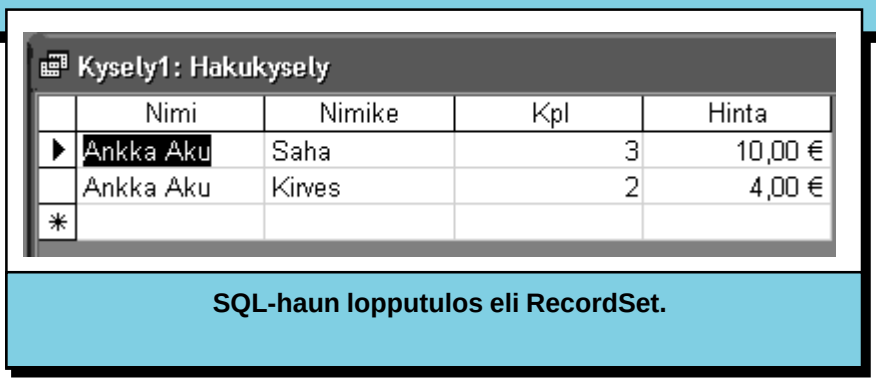
Sisäänrakennettu ACCESS-tuki helpottaa tietokantaohjelmoijan rutiineja tuntuvasti. Yleensä tiedot haetaan tietokannasta ACCESSista tutuilla SQL-lauseilla. Näin saatua tietojen joukkoa kutsutaan recordsetiksi - koska parempaa tapaa ei ole tullut vastaan, emme jatkossa puhu (SQL-haulla) löydettyjen tietojen joukosta taikka tietueesta, vaan recordsetistä. Kuvittele recordset samaksi moniriviseksi ACCESS-taulukoksi, jonka saat siellä valitsemalla aluksi

Kyselyt | Muodosta kysely rakennenäkymässä | SQL

ja tekemällä haun muodossa SELECT kentät FROM taulu.



SQL-kysely ACCESS-ohjelmassa. Mitä tuotteita Aku ankka on ostanut?



	Nimi	Nimike	Kpl	Hinta
▶	Ankka Aku	Saha	3	10,00 €
	Ankka Aku	Kirves	2	4,00 €
*				

SQL-haun lopputulos eli RecordSet.

Ajankohtaiseksi SQL-kyselyt tulevat , kun VB-ohjelmoija ottaa käyttöön Data(-object) tai ADO-rajapinnan tietokannan ja sovelluksen välille. Tuloksena SQL-hausta saadaan ns. RecordSet (tietuejoukko kyselyn toteuttavia tietoja).

Kyselyjä on kahdenlaisia: SELECT-lauseella saadaan uusi RecordSet ja INSERT jne- komennoilla muutetaan olemassaolevaa. Tässä muutamia SQL-esimerkkejä:

SELECT * From Asiakas;
-valitsee kaikki tiedot Asiakastaulukosta

SELECT Nimi,Paikkakunta From Asiakas INNER JOIN Posti On Asiakas.PNumero=Posti.Numero;

SELECT * FROM Asiakas WHERE Nimi LIKE “*Ankka*”
-etsitään kaikki tietokannan ankat
-joissakin ympäristöissä tulee käyttää %-merkkiä korvaamaan osajonoa

SELECT * FROM Asiakas WHERE Nimi LIKE “H*”
-etsitään kaikki tietokannan H-kirjaimella alkavat satuolennot

SELECT * From Asiakas ORDER BY PNumero [DESC]
-järjestetään tiedot [laskevaan] suuruusjärjestykseen

SELECT Nimi,Paikkakunta From Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero WHERE PNumero>=”11100" AND PNumero <=”15100"

VB:n sisäänrakennettu ACCESS-tuki olettaa tietokannan olevan Office 97-formaatissa. Office 2000 ja 2002 -versioissa tietokanta pitää muuttaa Työkalut tietokannan apuohjelmat | Muunna tietokanta -valinnalla Access97 -formaattiin. Myöhemmin ADO-rajapintaa käytettäessä huomataan, kuinka Microsoftin oma tietokantamoottori JETin avulla ohjelmat voivat hyödyntää myös Officen uudempia formaatteja.

Datakomponentilla luodaan edellytys tietokannan käyttöä varten. Datakomponentin ominaisuuksista DatabaseNamella määritellään levytiedostolla olevan tietokannan nimi ja RecordSource käytettävän taulukon(RecordSetin).

Muut sopivat Windows-komponentit käyttävät Datakomponenttia tietolähteenään (DataSource). Lisäksi on määritettävä se tietokenttä, minkä tiedot halutaan näkyviin (Datafield). Datakomponentti on hyvä ja helppo tapa käyttää ja ylläpitää ulkoisia tietovarastoja. Lisää toiminnallisuutta siihen saadaan määrittelemällä RecordSource ohjelmallisesti. Koska datakomponentti ei useinkaan sovi ulkonkönsä puolesta sovelluksiin, piilotetaan se. Tällöin liikkuminen tietueiden välillä tulee ohjelmoida itse.Seuraavassa on korvattu datakomponentin nuolet omilla nappuloilla

```
If Not (Data1.Recordset.EOF) Then
  Data1.Recordset.MoveNext
Else
  Data1.Recordset.MoveFirst
End If

If Not (Data1.Recordset.BOF) Then
  Data1.Recordset.MovePrevious
Else
  Data1.Recordset.MoveLast
End If
```

SQL-lauseen sisältämä kahden taulukon käyttöesimerkki

Data1.RecordSource=”SELECT Nimi,Paikkakunta FROM Asiakas INNER JOIN Posti WHERE PNumero>”15000"

SQL-lauseen oikea käyttö edellyttää Data-komponentin päivittämistä (Data1.Refresh)

Aliohjelmassa muodostetaan SQL-lauseella uusi RecordSet, jossa otetaan toinen tietokannan taulukoista käyttöön.

```
Private Sub Command1_Click()
 Data1.RecordSource = “SELECT * FROM Asiakas INNER JOIN Posti ON
 Asiakas.PNumero=Posti.Numero”
   Data1.Refresh
   Label3.Caption = Data1.Recordset.Fields(“paikkakunta”)
End Sub
```

Datakomponentin RecordSource voidaan määritellä myös Properties-kohdassa SQL-lauseena.

Esimerkki 18: Postinumerot

Tee ohjelma, jonka käynnistyessä viet kaikki postinumerot ComboBoxiin nimeltä pnumero ja valittuasi jonkun numeroista ohjelma tulostaa paikkakunnan.

0. Tee Access-tietokanta, jossa on kentät numero ja paikkakunta.

1. Tee Datakomponentti, jonka nimeät ja piilotat näkyvistä

.Name	***Posti***
.Visible	***false***

2. Tee Combobox

.Name	***PNumero***

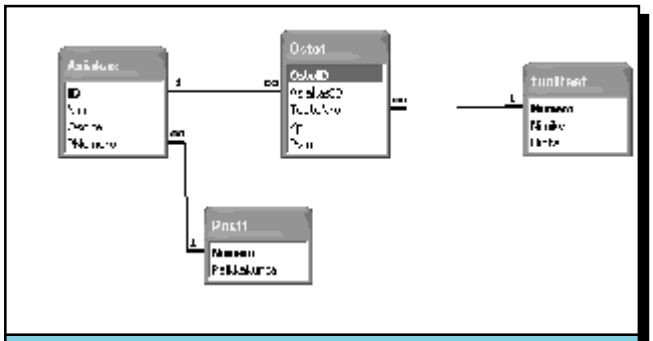
3. Tee tapahtumakäsittely Form_Load(), jossa lisäät comboboxin rivejä recordsetin alusta loppuun

```
Private Sub Form_Load ()
  Do Until Posti.Recordset.EOF
    PNumero.AddItem (Posti.Recordset.Fields(0))
    Posti.Recordset.MoveNext
  Loop
  PNumero.Text = “Valitse postinumero”
End Sub
```

4. Tee tapahtumakäsittely PNumero_Click(), jossa teet SQL-lauseen, jossa haet vain numerokentän tietoa vastaavia rivejä RecordSetistä.

```
Private Sub PNumero_Click()
Posti.RecordSource = “SELECT Paikkakunta,Numero From Posti WHERE Numero=’”& PNumero.Text & “‘“
Posti.Refresh
MsgBox (Posti.Recordset.Fields(“Paikkakunta”))
End Sub
```

14. SQL-harjoituksia



SQL-harjoitusten ACCESS-tietokannan yhteydet.

Taulukot yhdistetään toisiinsa SQL:n INNER JOIN -lauseilla seuraavan syntaksin mukaan:

SELECT [kentät] FROM taulu1 INNER JOIN taulu2 ON taulu1.kenttä=taulu2.kenttä WHERE ehto

Jos taulukoita on useampi kuin kaksi, käytetään sulkuja erottamaan yhteydet toisistaan. Tällöin sulkujen määrä on taulujen määrä miinus yksi. Huomaa, että eri SQL:n versioissa rakenne voi poiketa alla esitetystä.

Tee SQL-lause, jolla haet Aku Ankan ostokset (Nimi,Nimike,Kpl,hinta)

SELECT Nimi,Nimike, Kpl, Hinta FROM (Asiakas INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Nimi=”Ankka Aku”;

Tee haku, jossa haet kaikki ostot, joita on tehty enemmän kuin 1 kpl.

SELECT Nimi,Nimike, Kpl, Hinta FROM (Asiakas INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Kpl>1;

Tee haku, jossa luettelet ne henkilöt, jotka ovat tänään käyneet ostoksilla. Huomaa, että SQL-hauissa päivämää esitetään numeromerkkien sisällä jenkkimuodossa” eli kuukausi ensin. Lisäksi erottimena käytetään kauttaviivaa.

SELECT Nimi,Nimike, Pvm FROM (Asiakas INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Pvm=#7/1/2003#;

Tee haku, jolla selvität paljollako Aku osti tänään. Jos Format-lauseeseen ei voi kirjoittaa euron merkkiä, muista sen ASCII eli Chr(128).

SELECT Nimi, Format(kpl*hinta,”#####.00 “) & chr(128) AS [Ostot] FROM (Asiakas INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Pvm=#7/1/2003# AND Nimi=”Ankka Aku”;

Seuraavassa sama haku, mutta tulokset koottuna summa-funktion avulla yhdelle riville.

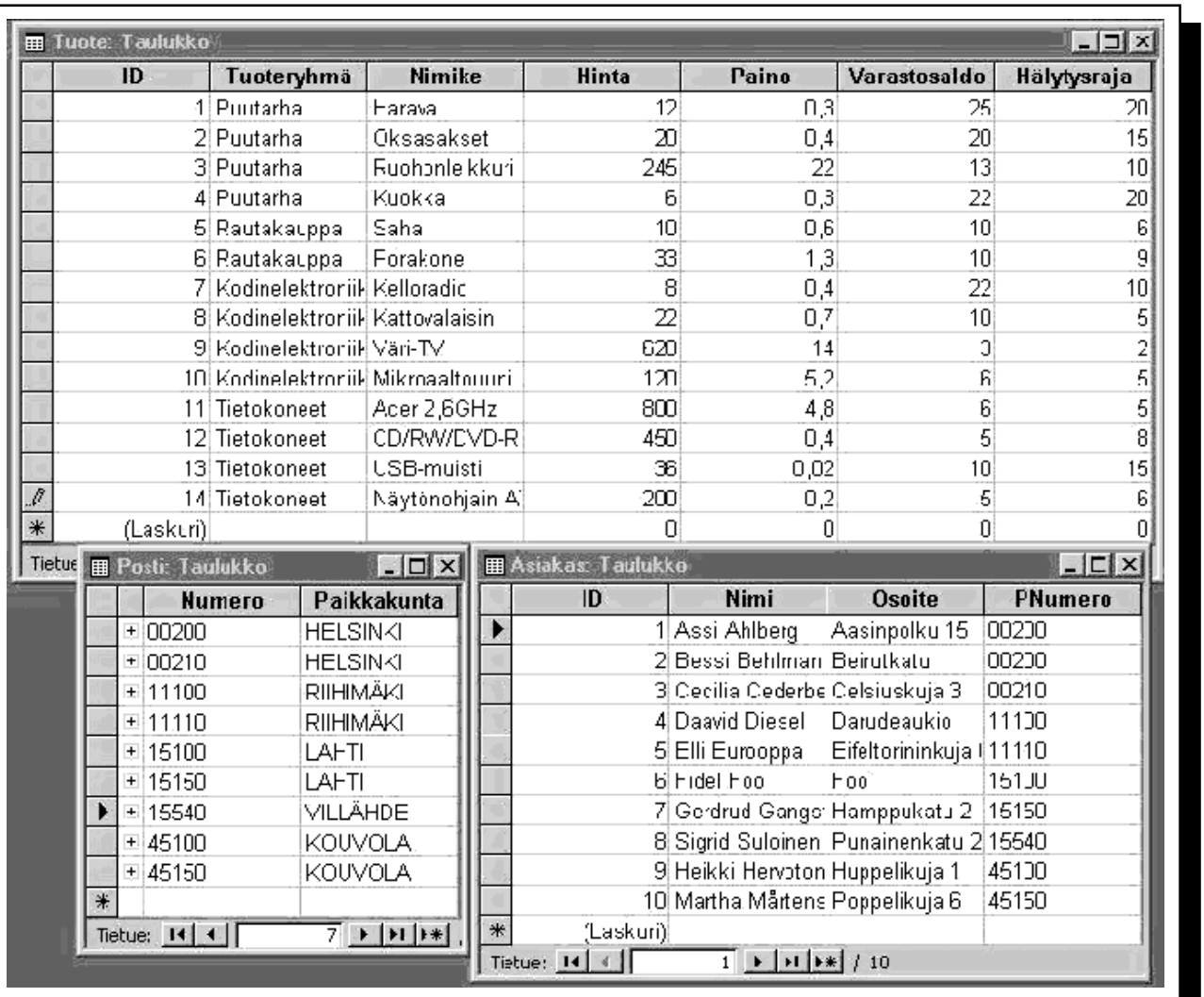
SELECT Nimi, Format(SUM(kpl*hinta),”####.00 •”) AS [Ostot] FROM (Asiakas INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Pvm=#7/1/2003# AND Nimi=”Ankka Aku” GROUP BY Nimi;

Kirjoita lause, jolla haet helsinkiläisiä, jotka ovat ostaneet tänään sahan. DISTINCT tulostaa vain yhden samaa tietoa vastaavan rivin.

SELECT DISTINCT Nimi, Nimike,Paikkakunta FROM ((Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero) INNER JOIN Ostot ON Asiakas.ID=Ostot.AsiakasID) INNER JOIN Tuotteet ON Ostot.TuoteNro=Tuotteet.Numero WHERE Paikkakunta="HELSINKI" AND Nimike="Saha";

Jatkoharjoitusten pohjaksi

Tee lomakkeelle data- eli tietokantaobjekti ja piilota se, koska se vain harvoin sopii sovelluksen ulkonäköön. Nimeä tietokantaobjekti tkannaksi. Tee tarvittaessa myös muita objekteja, jotka voit alla näkyvin esimerkein sitoa tietokantaan - tai pikemminkin tietokannasta SQL-kyselyn avulla saatuun recordsetiin.



SQL-harjoitusten esimerkkitietokanta, jossa on kolme taulukkoa: Asiakas, Posti ja Tuote.

Seuraavassa ohjelmakoodi, jolla avataan ACCESS-tietokanta nimeltä TKANTA97.MDB. Tällaisella lausekkeella voidaan korvata dataobjektin määrittelyt properties-ikkunassa. Ennen kuin mitään enempää voidaan tehdä, on muuttunut tietokanta päivitettävä (.Refresh). Samassa aliohjelmassa on viety recordsetin ensimmäinen postinumero tekstikenttään.

```
Private Sub avaa_Click()
  tkanta.DatabaseName = "\\president\visual\turunen\tkanta97.mdb"
  tkanta.RecordSource = "posti"
  tkanta.Refresh
  tieto.Text = tkanta.Recordset.Fields("numero")
End Sub
```

Tässä aliohjelmassa viedään järjestyksessä kaikki postinumerot ja paikkakunnat listaan. Toimenpide edellyttää, että tietokantaobjektin tiedostonimi, ja recordsource on määritelty aiemman esimerkin mukaisesti. Muista, että sekä tietokantaobjekti että lista voivat olla ohjelmassa piilotettuina (.visible=false) ja mahdolliset haut tapahtua täten käyttäjältä salassa.

```
Private Sub kaikki_Click()
Do Until tkanta.Recordset.EOF
  Lista.AddItem (tkanta.Recordset.Fields("Nimi"))
  Lista.AddItem (tkanta.Recordset.Fields("Osoite"))
  tkanta.Recordset.MoveNext
Loop
End Sub
```

Tietokannan taulukot voivat olla myös toisiinsa liitettyinä SQL-kielen INNER JOIN -komennoin. Esimerkin ACCESS-tietokannassa on määriteltynä yhden suhde moneen yhteys Asiakastaulukon postinumeron ja Postitaulukon postinumeron välille. Seuraavassa haetaan SQL-lauseella nimeä ja paikkakuntaa - huomaa SQL-lauseen lopussa olevat poikkeukselliset lainaus- ja heittomerkit (hakuavain on tekstikentässä). Normaalisti SQL-haun merkkijonot on lainausmerkeissä, mutta koska esimerkissä koko SQL-lause on jo lainausmerkeissä, joudutaan SQL-lauseen lainausmerkit korvaamaan heittomerkeillä. Tässä SQL-haku heittomerkein, jos tekstikentän sisältö olisi "Sigrid Suloinen":

SELECT nimi,paikkakunta FROM Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero
WHERE Nimi='Sigrid Suloinen'

Ensimmäinen lopun heittomerkeistä aloittaa haun, jossa käytetään aakkosnumeerista tietoa. Lopun lainausmerkeistä ensimmäinen lopettaa SQL-lauseen. Kun tekstikentän sisältö on liitetty &-merkeillä hakuun tulee lainausmerkki, joka tarvitaan SQL-haun aakkosnumeerisen tiedon lopun merkiksi kuuluvan heittomerkin liittämiseksi. Tällaiset haut ja tietojen yhdistäminen tulevat tutuksi ainoastaan harjoittelemalla. Jos et osaa muodostaa riittävän tarkkaa SQL-lausetta, hae tarvittava tieto recordsetiin ja loput haut voit tehdä siirtämällä tiedot Windows-objekteihin ja tekemällä haku niistä!

```
Private Sub hae_Click()
  tkanta.DatabaseName = "\\palvelin\visual\sql\tkanta97.mdb""
  tkanta.RecordSource=
 "SELECT nimi,paikkakunta FROM Asiakas INNER JOIN Posti " & _
 "ON Asiakas.PNumero=Posti.Numero WHERE Nimi='" & tieto.Text & "'"
 tkanta.Refresh
 tieto.Text = tkanta.Recordset.Fields("nimi") & " " & tkanta.Recordset.Fields("paikkakunta")
End Sub
```

Siis harjoituksiin...

..koska SQL-lauseet ovat jo lainausmerkeissä, tulee lauseen sisällä oleva aakkosnumeerinen tieto kirjoittaa heittomerkkien sisään.

Tulosta Puutarhatuoteryhmän kaikki tuotteet alekkain.

“SELECT Nimike,Tuoteryhmä FROM Tuote WHERE Tuoteryhmä=’Puutarha’”

Etsi ne tuotteiden nimikkeet, jotka painavat enemmän kuin neljä kiloa list-objektiin.

“SELECT Nimike,Paino FROM Tuote WHERE Paino>4”

Etsi ne tuotteet, joiden varastosaldo on pienempi kuin hälytysraja.

“SELECT Nimike,Varastosaldo,Hälytysraja FROM Tuote WHERE Varastosaldo>Hälytysraja”

Etsi kaikki K-kirjaimella alkavat tuotteet

“SELECT Nimike FROM Tuote WHERE Nimike LIKE ‘K*’”

Etsi ja lisää listaan, ne tuotteet joiden hinta on välillä 20• - 200•.

“SELECT Nimike,Hinta FROM Tuote WHERE Hinta>=20 AND Hinta<=200”

Mahtuuko kaikki puutarharyhmän tuotteet 25 kg:n rahtimaksun piiriin?

tkanta.RecordSource = “SELECT Nimike,Paino FROM Tuote WHERE Tuoteryhmä=’Puutarha’”

```
tkanta.Refresh
Lista.Clear
paino=0
Do Until tkanta.Recordset.EOF
  Lista.AddItem (tkanta.Recordset.Fields(“Nimike”) & “ “ &
tkanta.Recordset.Fields(“paino”))
paino=paino+tkanta.RecordSet.Fields(“paino”)
  tkanta.Recordset.MoveNext
Loop
MsgBox(paino)
```

Mitkä tuotteet varmuudella hälyttävät varastossa seuraavan kolmen saman tuotteen oston yhteydessä?

“SELECT Nimike FROM Tuote WHERE Varastosaldo-Hälytysraja<=3”

Ketkä asiakkaista asuvat Helsingissä?

“SELECT Nimi FROM Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero WHERE Paikkakunta=’HELSINKI’”

Tulosta Helsinkiä pohjoisempana asuvat asiakkaat?

“SELECT Nimi FROM Asiakas WHERE PNumero>’003’”

Ketkä Helsingissä asuvista ovat aakkosissa ennen Ceciliaa?

“SELECT Nimi FROM Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero WHERE Paikkakunta=’HELSINKI’ AND Nimi<‘Cecilia’”

Keiden katuosoitteesta löytyy kakkonen?

“SELECT Nimi, Osoite FROM Asikas WHERE Osoite LIKE ‘*2*’”

Ketkä asiakkaat eivät pääse autolla kuin yhtä reittiä pois kotoaan?

“SELECT Nimi, Osoite FROM Asikas WHERE Osoite LIKE ‘*kuja*’”

Mitä Tietokoneet-ryhmän tuotteet maksavat ACER-tietokonetta lukuunottamatta?

“SELECT Nimike,Hinta FROM Tuote WHERE Tuoteryhmä=’Tietokoneet’ AND Not(Nimike LIKE ‘ACER*’)”

Monta tietuetta tuoterekisterissä on?

“SELECT Nimike FROM Tuote”
```
tkanta.RecordSet.MoveLast
MsgBox(tkanta.RecordSet.RecordCount)
```

For Each-toisto on kätevä tapa käydä läpi taulukkomaisia rakenteita, varsinkin jos ei tiedä niiden alkioiden nimiä tai määrää. Seuraavassa esimerkissä käydään SQL-lausekkeella saadun recordsetin kaikki alkiot läpi - ei pelkästään yhden taulukon tietoja.

```
Private Sub Command1_Click()
tkanta.DatabaseName = “\\PALVELIN\VISUAL\SQL\tkanta97.mdb”
tkanta.RecordSource = “SELECT * FROM Asiakas INNER JOIN Posti ON
Asiakas.PNumero=Posti.Numero”
tkanta.Refresh
Lista.Clear
Do Until tkanta.Recordset.EOF
  For Each Field In tkanta.Recordset.Fields
    Lista.AddItem (Field.Name & “: “ & Field)
  Next Field
  tkanta.Recordset.MoveNext
Loop
End Sub
```

Esimerkki 19: Tuoterekisteri

Tee sovellus, jossa viet koko tuoterekisterin nimikkeet comboboxiin. Tee comboboxiin Click()-tapahtuma, jossa saat comboboxin text-kentästä SQL-lauseen ehdon hintahaulle. Samalla lasket valittujen tuotteiden hintoja ja tulosta nimike, hinta ja yhteishinta MsgBoxissa.

```
Dim summa

Private Sub avaa_Click()
  Nimi.Clear
  tkanta.DatabaseName = “\\PALVELIN\VISUAL\SQL\tkanta97.mdb”
 tkanta.RecordSource = “SELECT Nimike FROM Tuote”
  tkanta.Refresh
    Do Until tkanta.Recordset.EOF
    Nimi.AddItem (tkanta.Recordset.Fields(“nimike”))
    tkanta.Recordset.MoveNext
  Loop
  tkanta.Recordset.Close
  Nimi.Text=”Valitkaa tuote – olkaa hyvä!”
  summa = 0
End Sub
```

```
Private Sub Nimi_Click()
tkanta.DatabaseName = "\\PALVELIN\VISUAL\SQL\tkanta97.mdb"
tkanta.RecordSource = "SELECT Nimike,Hinta FROM Tuote WHERE Nimike='" &
Nimi.Text & "'"
tkanta.Refresh
summa = summa + tkanta.Recordset.Fields("Hinta")
MsgBox (tkanta.Recordset.Fields("Nimike") & " " &
tkanta.Recordset.Fields("Hinta") & " YHT. " & summa)
End Sub
```

Esimerkki monimutkaisemmasta SQL-hausta, jossa haetaan kaikkia ostosrivejä vastaavia tietoja.

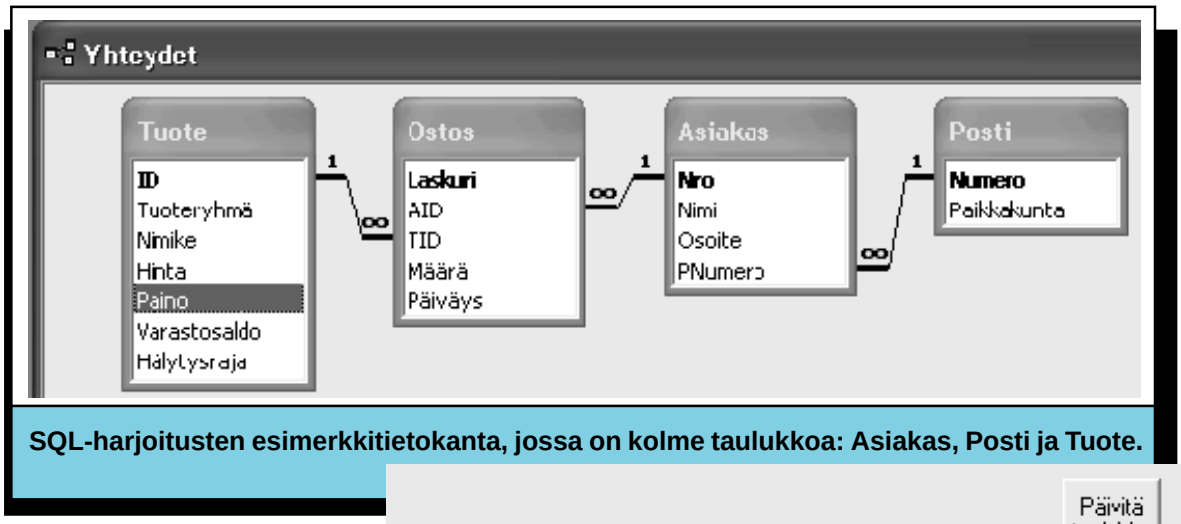
SELECT * FROM (((Asiakas INNER JOIN Posti ON Asiakas.PNumero=Posti.Numero) INNER JOIN Ostos ON Asiakas.Nro=Ostos.AID) INNER JOIN Tuote ON Ostos.TID=Tuote.ID);

Tietokannan muuttaminen (AddNew, Edit)

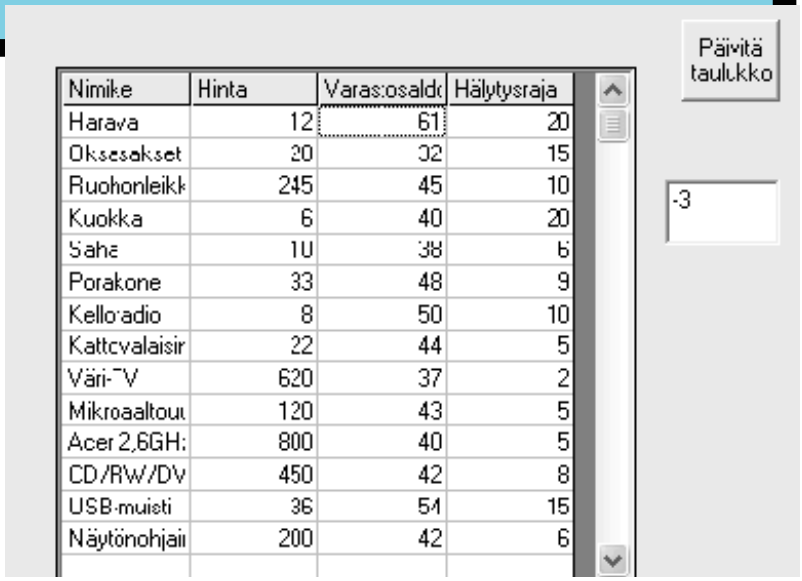
Tietokentän muuttaminen vaatii tietokannan recordsetin alustamista Edit- ja tietueiden lisääminen AddNew-komennolla. Kun muutokset on tehty, recordset päivitetään UpDate-komennolla.

Esimerkki 19: Kassa-varastodemo

Tee sovellus, jossa ostettaessa tuotteita, niiden hinta katsotaan tietokannasta ja varastosaldo pienenee ostetulla määrällä. Jos varastosaldo pienenee alle hälytysrajan, ilmoitetaan siitä erikseen.



SQL-harjoitusten esimerkkitietokanta, jossa on kolme taulukkoa: Asiakas, Posti ja Tuote.



```
Private Sub Command1_Click()
  taulu.Clear
  tkanta.DatabaseName = "\\PALVELIN\VISUAL\SQL\tkanta97.mdb"
 tkanta.RecordSource = "SELECT Nimike,Hinta,Varastosaldo,Hälytysraja FROM Tuote"
  tkanta.Refresh
  taulu.TextMatrix(0, 0) = "Nimike"
  taulu.TextMatrix(0, 1) = "Hinta"
  taulu.TextMatrix(0, 2) = "Varastosaldo"
  taulu.TextMatrix(0, 3) = "Hälytysraja"
  Do Until tkanta.Recordset.EOF
    taulu.TextMatrix(taulu.Rows - 1, 0) = tkanta.Recordset.Fields("Nimike")
    taulu.TextMatrix(taulu.Rows - 1, 1) = tkanta.Recordset.Fields("Hinta")
    taulu.TextMatrix(taulu.Rows - 1, 2) = &_
    tkanta.Recordset.Fields("Varastosaldo")
    taulu.TextMatrix(taulu.Rows - 1, 3) = &_
    tkanta.Recordset.Fields("Hälytysraja")
    taulu.Rows = taulu.Rows + 1
    tkanta.Recordset.MoveNext
  Loop
  tkanta.Recordset.Close
End Sub
```

Yhden tuotteen haku ja Ostos-taulukon päivittäminen uudella rivillä

```
Private Sub taulu_Click()
Dim TuoteID
  tkanta.DatabaseName = "\\PALVELIN\VISUAL\SQL\tkanta97.mdb"
  tkanta.RecordSource = "SELECT ID,Nimike,Hinta,Varastosaldo,Hälytysraja FROM  Tuote WHERE
Nimike='" & taulu.TextMatrix(taulu.Row, 0) & "'"
  tkanta.Refresh
  TuoteID = tkanta.Recordset.Fields("ID")
  If Val(osto.Text) <= tkanta.Recordset.Fields("Varastosaldo") Then
    If tkanta.Recordset.Fields("Hälytysraja") >= &_
    tkanta.Recordset.Fields("Varastosaldo") - Val(osto.Text) Then
      MsgBox ("Tilaa tuotetta lisää!")
    End If
    tkanta.Recordset.Edit
    tkanta.Recordset.Fields("Varastosaldo") = &_
    tkanta.Recordset.Fields("Varastosaldo") - Val(osto.Text)
    tkanta.Recordset.Update
    Command1_Click
    tkanta.DatabaseName = "\\PALVELIN\VISUAL\SQL\tkanta97.mdb"
    tkanta.RecordSource = "Ostos"
    tkanta.Refresh
    tkanta.Recordset.AddNew
    tkanta.Recordset.Fields("AID") = 1
    tkanta.Recordset.Fields("TID") = TuoteID
    tkanta.Recordset.Fields("Määrä") = osto.Text
    tkanta.Recordset.Fields("Päiväys") = Now
    tkanta.Recordset.Update
    tkanta.Recordset.Close
  End If
End Sub
```

Kehittele sovellusta siten, että saat asiakastiedot toisella Dataobjektilla selattavaksi. Älä piilota tietokantaominaisuutta, vaan yhdistä se kaikkiin asiakkaan tietoihin omilla tekstikentillään. Siirrä asiakkaan ID-numero edelliseen ohjelmakoodiin. Lisäksi sovelluksessa tulee olla listaobjekti, johon kerätään ostotapahtumat hinnat laskettuina. Kun asiakas vaihtuu tyhjenee lista.