Delphi pradmenys

Parengė
Vaidilutė Žukauskienė
Informacinių technologijų mokytoja metodininkė

Turinys

1. Objektai

Objektinis programavimas vartoja naują duomenų tipą – **objektą**. Jis gaunamas jungiant duomenis su tuos duomenis apdorojančiomis procedūromis.

Objektą nusako duomenys, jie *Delphi* sistemoje vadinami **savybėmis** arba **parametrais** (angl. *properties*). Savybė – tai dydis, spalva, padėtis ir pan.

Procedūros, galinčios apdoroti su objektu ir jo savybėmis susijusią informaciją, vadinamos **metodais** arba **įvykiais** (angl. *events)*. Pavyzdžiui, lango objekto metodai leidžia langą kilnoti, keisti jo dydį, piešti ir pan.

Mes naujų objektų nekursime, o keisime jau esamų tipinių objektų savybes, pritaikydami savo reikmėms.

Delphi vaizdiniai objektai – tai **formos** ir **standartiniai komponentai.** Formomis vadinami langai, kuriuose talpinami standartiniai komponentai: etiketės, mygtukai, meniu ir pan. Programa gali turėti daug formų.

Objektų naudojimo sintaksė panaši į *Pascalio* kalbos įrašų sintaksę. Objekto ir savybės vardą skiria taškas. Pavyzdžiui, jei objekto „Button1“ savybei „Width“ norime priskirti konkrečią reikšmę, tai rašome:

```
Button1.Width := 100;
```

Objekto vardą nuo jo metodo taip pat skiria taškas – norėdami parodyti ekrane formą Form3 rašome

```
Form3.Show;
```

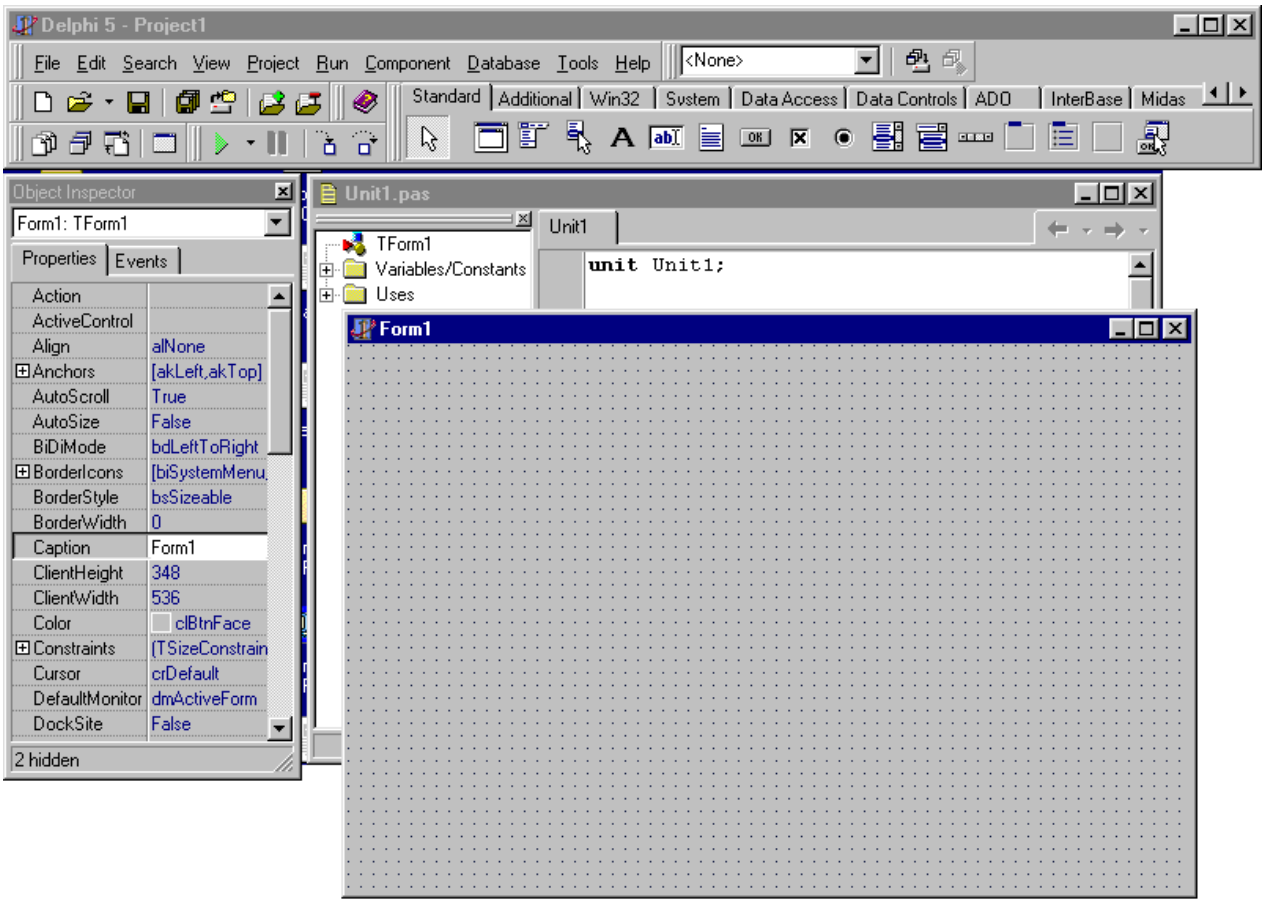
2. Valdymas įvykiais

Delphi remiasi valdymu įvykiais (angl. *events*). Įvykis – tai veiksmas, kurį atpažįsta ir į kurį gali reaguoti programos objektai. Įvykius gali inicijuoti ir vartotojas (pavyzdžiui, paspausdamas pelės mygtuką ar klaviatūros klavišą), ir pati programa (pavyzdžiui, atsiradus kuriai nors klaidai, gavus programos chronometro signalą). Objektai (formos bei komponentai), reaguodami į įvykį, vykdo programos dalis – įvykių procedūras, kurias rašome mes. Įvykiai atpažįstami automatiškai, nes tuo rūpinasi pati *Delphi* sistema. Pavyzdžiui, spustelėjus pelės mygtuką, kai jos žymeklis yra ant mygtuko, įvyksta mygtuko paspaudimo (angl. *click*) įvykis.

Valdant įvykiais, objektai yra lyg savarankiški veikėjai, kuriems nustatyta, kaip reaguoti į vieną ar kitą įvykį. Tuo tarpu visas veikimo scenarijus sunkiai prognozuojamas, nes pagrindinis režisierius yra vartotojas. Esant didžiulei objektų, įvykių rūšių bei vartotojo veiksmų įvairovei tradiciškai kurti sudėtingas programas yra pernelyg keblu. Tuo tarpu valdant įvykiais kurti sudėtingą programinį produktą yra paprasčiau, nes kuriamos tiesiog atskirų įvykių procedūros. Jei įvykių, į kurios objektai moka reaguoti, nėra, programa tiesiog laukia kito įvykio.

3. Delphi sąsaja

Delphi 5 versijos grafinės sąsajos sandara (pav. 1):



pav. 1

Vaizdiniai komponentai – standartinių komponentų rinkiniai (paletės).

Kuriamos programos lango šablonas (forma) – programos sąsajos projektavimui skirtas šablonas, į kurį perkeliami vaizdiniai komponentai. Užrašą *Form1* galima keisti, taip pat galima

keisti ir formos dydį ir pan. Keitimus galima daryti rankiniu būdu projektuojant ir programos vykdymo metu.

Objektų inspektorius – vaizdinių komponentų savybių (angl. *Properties*) ir jų kontroliuojamų įvykių (angl. *Event)* parinkimo kortelės. Savybių dalyje galima rankiniu būdu nustatyti objektų savybes (pavyzdžiui: suteikti formai vardą, nustatyti jos dydį). Įvykių – pasirinkti įvykį, į kurį tas objektas turi reaguoti. Ant atitinkamo įvykio (pavyzdžiui, prie **OnClick,** t. y. kai bus paspaustas objektas) du kartus spustelėję kairįjį pelės mygtuką, atsivers programos redagavimo langas, kuriame galime rašyti programą tam įvykiui.

Projekto moduliai – moduliai, iš kurių sudaroma Delphi programa (projektas). Šioje vietoje rašomos komandos, kurios bus vykdomos įvykus vienam ar kitam to objekto įvykiui.

4. Pirmoji programa. Etiketės. Mygtukai.

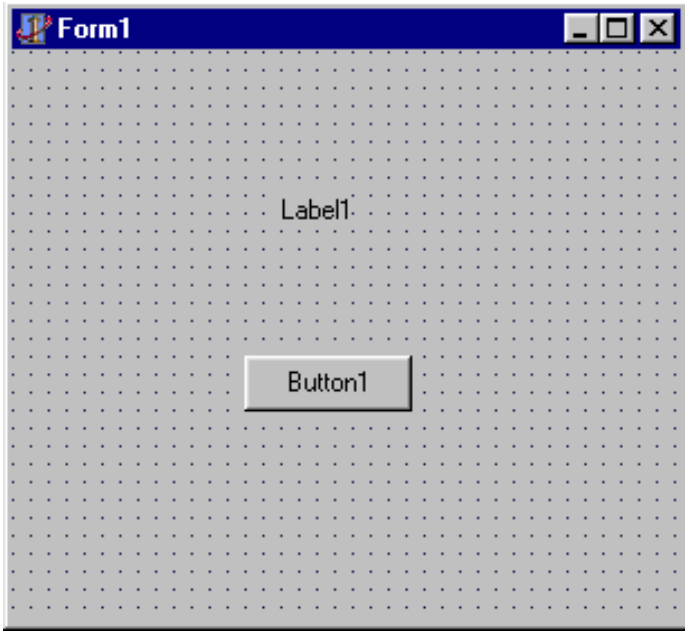
Sukursime programą, kuri pateiktų langą su vienu mygtuku. Jį paspaudus lange pasirodytų pasveikinimas „Sveiki!“.

Formoje padedame etiketę **Label1**, kurioje bus parašytas žodis „Sveiki!“ bei mygtuką **Button1**.

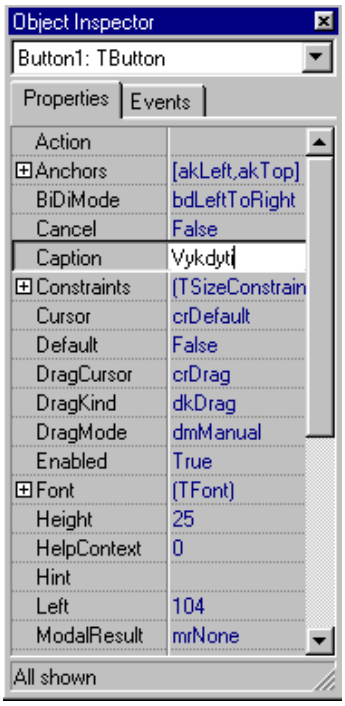
Standartinių komponentų paletėje pasirenkame mums reikalingą objektą užvesdami pelės rodyklę ant to objekto ir paspausdami kairįjį mygtuką. Objektą įkeliame į formą atitinkamoje formos vietoje paspaudę pelės kairįjį mygtuką.

Formoje (**Klaida! Nerastas nuorodos šaltinis.**) turime etiketę **Label1** ir mygtuką **Button1** (skaičius „1“ prie žodžio „Label“ („Button“) reiškia, kad tai pirmoji etiketė (mygtukas) formoje, jei formoje būtų dar kelios etiketės (mygtukai), tai prie Label (Botton) būtų atitinkamai skaičiai 2, 3, t.y. Edit2, Edit3 ir t.t.).

Dabar mygtuko užrašą „Button1“ pakeisime į „Vykdyti“. Tam objektų inspektoriuje ties mygtuko savybe **Caption** parašysime „Vykdyti“ (**Klaida! Nerastas nuorodos šaltinis.**). Norėdami, kad objektų inspektoriuje būti matomos mygtuko savybės, turime formoje aktyvuoti mygtuką (t.y. paspausti pelės kairįjį mygtuką ant mums reikalingo objekto). O norėdami etiketėje panaikinti tekstą „Label1“, aktyvuojame ją ir savybėje **Caption** paliekame tuščią langelį. Formos pavadinimą taip pat analogiškai pakeičiame į „Pirma programa“. Jei norime, kad užrašas atrodytų gražiau, galime objektų inspektoriuje pakeisti etiketės savybę **Font.**



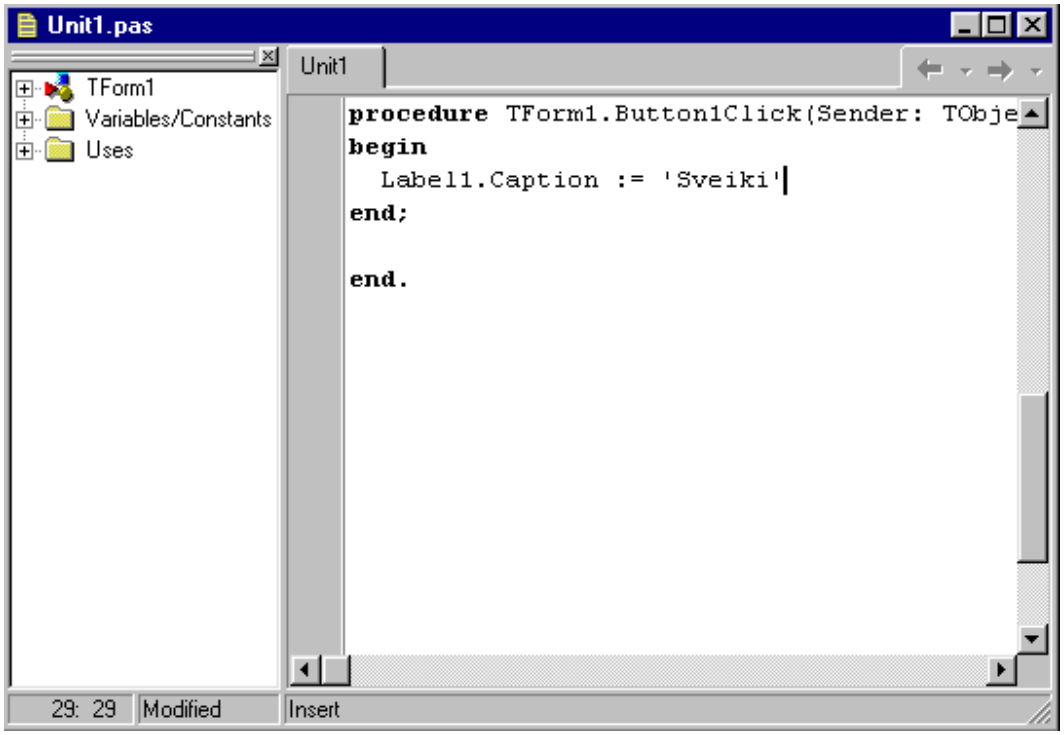
pav. 2



pav. 3

Dabar kursime mygtuko Button1 įvykį. Tam objektų inspektoriuje pasirenkame įvykių dalį ir teis įvykiu **OnClick** du kartus paspaudžiame kairįjį pelės mygtuką. Mums atsiveria programos redagavimo langas su iš karto atsiradusia įvykio procedūra (objekto įvykio **OnClick** procedūra atsivers ir du kartus paspaudus kairįjį pelės mygtuką ant paties objekto), čia rašome priskyrimo sakinį (pav. 4).

Norėdami vykdyti parašytą programą, pasirenkame meniu komandą **Run** ir atsidariusiame komandų sąraše renkamės komandą **Run** arba skaudžiame funkcinį klavišą **F9**. Taip pat galime pasirinkti mygtuką Run  mygtukų juostoje. Programos vykdymo metu paspaudę mygtuką „Vykdyti“ matysime kažką panašaus kaip pav. 5. Čia etiketės Label1 savybė Font pakeista – padidintas simbolių aukštis iki 24, šriftas parinktas Times New Roman.



pav. 4

pav. 5

Kadangi nenumatėme, kaip užbaigti programą, tai ją tenka uždaryti kaip ir visas *Windows* programas.

SVARBU!

Siekiant apsisaugoti nuo kelių programų projektų elementų sumaišymo rekomenduojam kiekvienam projektui sukurti atskirą katalogą ir visus projekto elementus rašyti tik į jį.

PRAKTINĖ UŽDUOTIS

- Papildykite formą dar vienu mygtuku, kurį paspaudus vietoj užrašo „Sveiki!“ atsirastų užrašas „Viso gero!“.
- Papildykite formą trečiu mygtuku, kurį paspaudus, programa baigtų darbą. Patarimas: trečiojo mygtuko procedūra atrodys taip:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Close
end;
```

5. Spalvinės objektų savybės

Daugelis objektų turi spalvines savybes: fono spalvą bei teksto ar piešinio.

Projektuojant objekto spalva keičiama objektų inspektoriuje pasirenkant savybės **Color** reikšmes iš išsiskleidžiančio sąrašo.

Norėdami keisti objekto spalvą vykdydami programą, spalvą galime nurodyti žodžiais (anglų kalba), tik prieš spalvos pavadinimą reikia pridėti priešdėlį ***cl***. Pavyzdžiui, norėdami objektą nudažyti raudona spalvą turime rašyti:

```
Objektas.Color := clRed;
```

Populiariausių spalvų lentelė

1. clAqua	skaisčiai žydra
2. clBlack	juoda
3. clBlue	mėlyna
4. clDkGray arba clGray	pilka
5. clFuchsia	skaisčiai violetinė
6. clGreen	žalia
7. clLime	Skasčiai žalia
8. clLtGray	Šviesiai pilka
9. clMaroon	Tamsiai raudona

10. clNavy	Tamsiai mėlyna
11. clOlive	Samanų spalva
12. clPurple	Tamsiai violetinė
13. clRed	Skaisčiai raudona
14. clSilver	Sidabrinė
15. clTeal	Murzinai melsva
16. clWhite	Balta
17. clYellow	Geltona

Taip pat patogiau vartoti RGB funkciją, turinčią 3 argumentus, atitinkančius pagrindinių spalvų (raudonos, žalios, mėlynos) intensyvumą. Intensyvumas kinta nuo 0 iki 255. Pavyzdžiui anksčiau pateiktą pavyzdį galėtume užrašyti taip: Objektas.Color := RGB(255, 0, 0).

Tačiau čia pateiktas pavyzdys tiktų tik fono spalvos keitimui, o jei norėtume pakeisti teksto spalvą, tai turėtume dar nurodyti, kad tai bus būtent teksto spalva, pavyzdžiui norėdami padaryti etiketės Label1 foną mėlyną, o teksto spalvą – raudoną rašome:

```
Label1.Color := clBlue;
Label1.Font.Color := clRed;
```

arba

```
Label1.Color := RGB(0, 0, 255);
Label1.Font.Color := RGB(255, 0, 0);
```

PRAKTINĖ UŽDUOTIS

- Sukurkite programą, kurios lange būtų trys langa spalvą keičiantys mygtukai.

6. Geometrinės objektų savybės

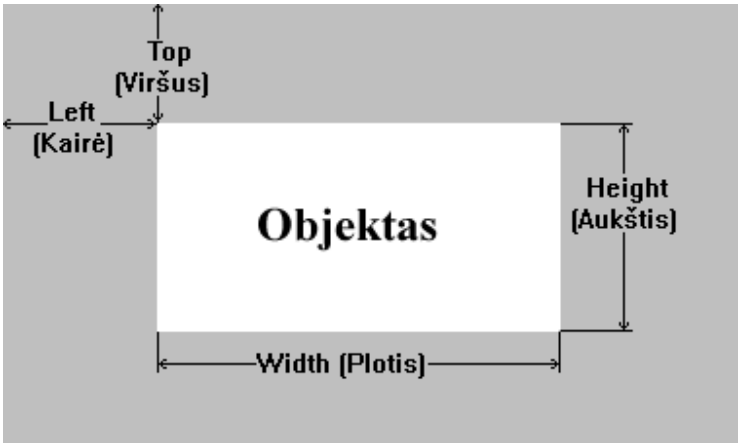
Beveik visi *Delphi* sistemos objektai turi padėtį ir dydį nusakančias savybes (pav. 6):

Padėties savybės: *Left* (kairė) ir *Top* (viršus).

Dydžio savybės: *Width* (plotis) ir *Height* (aukštis).

Paprastai pradinės geometrinių savybių reikšmės nustatomos automatiškai, projektavimo metu pele keičiant objektų dydį ir padėtį. Keisdami šias savybes programai veikiant, galėsime judinti komponentus, keisti jų dydį.

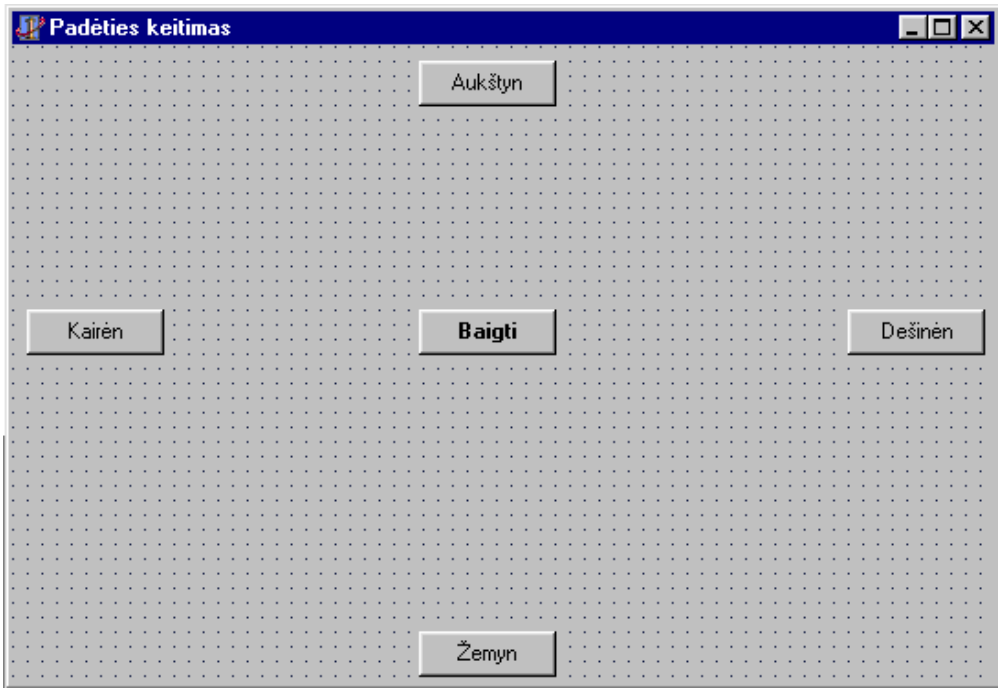
Koordinačių sistemos pradžia formai – kairysis viršutinis ekrano kampas. Bet kuriam tos formos komponentuii koordinačių sistemos pradžia – kairysis viršutinis formos kampas.



pav. 6

PRAKTINĖ UŽDUOTIS

- Sukurkite programą, kurios lango kraštuose būtų keturi mygtukai – „Aukštyn“, „Žemyn“, „Kairėn“ ir „Dešinėn“ – „traukiantys prie savęs“ formos viduryje esantį mygtuką „Baigti“. Spaudžiant mygtuką „Aukštyn“, mygtukas „Baigti“ turi kilti aukštyn, o spaudžiant mygtuką „Žemyn“ – leistųsi žemyn ir t.t. Paspaudus mygtuką „Baigti“, programa turi baigti darbą. Suprojektuota programos forma turėtų atrodyti taip (pav. 7):



pav. 7

- Sukurkite programą, kurios lange būtų trys mygtukai „Didėk“, „Mažėk“ ir „Baigti“. Spaudžiant mygtuką „Didėk“, mygtukas „Baigti“ turi didėti (į plotį ir aukštį). Spaudžiant mygtuką „Mažėk“, mygtukas „Baigti“ turi analogiškai mažėti. Paspaudus mygtuką „Baigti“, programa turi baigti darbą.

7. Objektų matomumas ir veiklumas

Tada, kai reikia paslėpti ar padaryti matomu paslėptą objektą, naudojama savybė *Visible* (matomumas). Tai yra loginio tipo kintamasis – kai jo reikšmė *True*, objektas yra matomas, o kai *False* – nematomas.

Kai norime kokį nors objektą padaryti neveikiančiu, nes jo veikimas tuo metu būtų beprasmis, naudojama savybė *Enabled* (veiklus), kurios reikšmė padaroma *False* (paprastai ji būna *True*). Pavyzdžiui, mygtukas, parodantys užrašą yra beprasmis, kai užrašas jau matomas.

PRAKTINĖ UŽDUOTIS

- Sukurkite programą, kurios lange būtų du mygtukai: „Rodyti“ ir „Slėpti“. Paspaudus mygtuką „Rodyti“ formoje pasirodytų koks nors užrašas, o paspaudus „Slėpti“, užrašas dingtų. Kai užrašas matomas, mygtukas „Rodyti“ turi būti neveiklus. Kai užrašas paslėptas, mygtukas „Slėpti“ turi būti neveiklus. Programos vykdymo pradžioje užrašas turi būti nematomas, mygtukas mygtukas „Rodyti“ – veiklus, mygtukas „Slėpti“ – neveiklus.
  Patarimas: pradinės programos sąlygas nusakomos formos įvykiu **OnActivate**.

8. Teksto įvedimas naudojant Edit laukelį

Objektas *Edit* skirtas vienos eilutės tekstui įvesti arba išvesti. Tam naudojama savybė *Text*. Tekstinis eilutė *Edit* turi įdomią savybę *PasswordChart* – slaptažodžio simbolis. Jei šios savybės reikšmė yra ne *#0*, o kitas, pavyzdžiui ’*’, tai programos vykdymo metu rašant tekstą lauke, parašyto teksto simboliai bus pakeisti ’*’ simboliu.

PRAKTINĖ UŽDUOTIS

- Sudarykite galimybę taip įvesti tekstą, kad berenkant jis iš karto automatiškai kartotųsi trijose formos vietose, skirtingais šriftais, dydžiais ir spalvomis.

Patarimas. Sukurkite teksto lauką tekstui įvesti ir kelias etiketes įvestam tekstui kartoti. Panaudokite teksto laukui įvykį *OnChange* (pasikeitimas), kuriam parašykite įvykio procedūrą.

- Formoje yra tekstinė eilutė, etiketė ir mygtukas. Tekstiniame lauke įvedame slaptažodį ir spaudžiame mygtuką. Jei slaptažodis teisingas, tai žymėje atsiranda pranešimas „Slaptažodis teisingas“, priešingu atveju – „Slaptažodis neteisingas“. Slaptažodis sugalvojamas programos kūrimo metu. Vartotojas jo pakeisti negali.

  Patarimas. Tekstinio lauko savybėje *PasswordChar* reikia padėti žvaigždutę ’*’.

9. Paveikslėlių panaudojimas

Paveikslo komponentas  yra „Additional“ paletėje. Paveikslas – objektas, kuriame galima piešti ir į kurį galima įkelti paveikslėlį iš grafinio tipo failo (*.bmp, *.ico, *.emf, *.wmf). Tam naudojama savybė **Picture** formos projektavimo režime arba programiškai. Naudojant *Picture* savybę, galima įkelti paveikslėlį ir iš *.jpg tipo failo, tačiau jei planuojama paveikslėlį įkeltį programos vykdymo metu, reikia *.jpg tipo failą konvertuoti į *.bmp. Programiškai paveikslėlis įkeliamas naudojant procedūrą **`Image1.Picture.LoadFromFile(`***'Kelias iki bylos'***`).`**

Savybė *Strech* nustato, ar įkeltas paveiksliukas bus tokio pat dydžio kaip ir paveikslas.

PRAKTINĖ UŽDUOTIS

- Sukurkite programą, kurios lange yra paveikslėlis. Paveikslėlį galima didinti ir mažinti spaudinėjant mygtukus „Didinti“ ir „Mažinti“.
- Sukurkite programą, kurios lange būtų trys mygtukai, kiekvieną iš jų paspaudus pasirodytų vis kitas paveikslėlis, o po paveikslėliu – jo komentaras.

10.Skaičių įvedimas ir išvedimas

Programos ryšys su jos lango komponentais palaikomas tik tekstinio (**string**) tipo duomenimis, o skaičiavimus galima atlikti tik su skaičiams skirtų tipų reikšmėmis (**integer, real**). Todėl redaguojamo lauko reikšmę iš pradžių reikia pertvarkyti iš teksto į skaičių ir tik po to atlikti skaičiavimus. Tokį pertvarkymą atlieka funkcijos **StrToInt** (tekstas keičiamas į sveikojo tipo skaičių) ir **StrToFloat** (tekstas keičiamas į realiojo tipo skaičių). Skaičiavimo rezultatą (skaičių)

norint pateikti etiketėje, jį reikia pertvarkyti į tekstinę formą. Realiuosius skaičius taip pertvarko funkcija **FloatToStr**, o sveikuosius – funkcija **IntToStr**.

PRAKTINĖ UŽDUOTIS

- Sudarykite programą, kuri skaičiuotų, kiek reikia mokėti už elektrą, kai duoti elektros skaitiklio parodymai „Iki“ ir „Nuo“. Bei žinome, jog tarifas yra 0,29 Lt.

11.Kritinių situacijų kontrolė ir pranešimai

Ši tema glaudžiai susijusi su skaičiavimo uždavinių programavimu. *Delphi* sistemoje yra numatyta apsauga nuo klaidingo programos argumentų įvedimo. Situacija, kai programos darbo metu negalima tęsti programoje numatytų veiksmų, vadinama kritine. Programuotojo numatyta reakcija į kompiuterio kontroliuojamą kritinę situaciją gali būti aprašoma tokia sintaksine struktūra:

try

<Apsaugomų sakinių sarašas>

except

<Reakcijos į kritinę situaciją aprašymas>

end;

Pranešimui formuoti naudojame procedūrą **ShowMessage**. Standartinė procedūra **Exit** nutraukia procedūros darbą, aptikus kritinę situaciją.

Tikrinant operatorių **try** Delphi aplinkoje, reikia išjungti automatinį programos nutraukimą kritinėse situacijose.Tam renkamės meniu komandą **Tools**. Atsidariusiame komandų sąraše pasirenkame **Debuger Options**. Atsidariusiame dialogo lange reikia išjungti integruotos aplinkos klaidų paiešką – panaikinti žymą **Integrated Debugging.**

PAVYZDYS:

Sudarykime programą, ištraukti kvadratinei šakniai, kurios argumentas yra įvedamas programos lango redaguojamame lauke. Programa turi tikrinti, ar nėra į laukelį įrašyto teksto keitimo skaičiumi klaida, o taip pat ar reikšmė yra neneigiamas skaičius.

```
procedure TForm1.Button1Click(Sender: TObject);
  var sk, saknis : real;
begin
  try                    // bandom atlikti veiksmus
    sk := StrToFloat (Edit1.Text);
    saknis := Sqrt ( a );
  except                 // rašom, ką daryti, jei nepavyko
  ShowMessage ( ' Įvedimo klaida ' );
  Edit1.clear;
  Exit;
end;
  Label4.caption := FloatToStrF (saknis, ffFixed, 7, 2);
end;
```

PRAKTINĖ UŽDUOTIS.

- Sudarykite programą, kuri rastų trikampio plotą, kai duotos visos trys kraštinės.

12.Jungikliai

Jungikliai (angl. **CheckBox**, piktograma) skirti kokiam nors režimui įjungti arba išjungti, keisti būklei ir pan. Jungikliai veikia nepriklausomai vienas nuo kito. Jungiklio būklę rodo savybės **Checked** reikšmė. Jei mygtukas įjungtas, tai ji lygi *True*, jei išjungtas – *False*.

PAVYZDYS:

Sukurkime formą su keturiais jungikliais (pav. 8), keičiančiais etiketės šriftą (šriftas vienu metu gali būti visų keturių tipų).

pav. 8

Jungiklių įvykių procedūros bus tokios:

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked
    then Label1.Font.Style := Label1.Font.Style + [fsBold]
    else Label1.Font.Style := Label1.Font.Style - [fsBold]
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
  if CheckBox2.Checked
    then Label1.Font.Style := Label1.Font.Style + [fsItalic]
    else Label1.Font.Style := Label1.Font.Style - [fsItalic]
end;

procedure TForm1.CheckBox3Click(Sender: TObject);
begin
  if CheckBox3.Checked
    then Label1.Font.Style := Label1.Font.Style + [fsUnderline]
    else Label1.Font.Style := Label1.Font.Style - [fsUnderline]
end;
```

PRAKTINĖ UŽDUOTIS.

Sudarykite programą, kurios lange surašyti patiekalų pavadinimai ir jų kainos. Taip pat yra etiketė, kurioje galima matyti, kiek kainuoja visi vartotojo pasirinkti patiekalai. Šalia kiekvieno patiekalo yra jungiklis, kurį paspausdamas vartojas gali nurodyti, ar jis renkasi patiekalą.

13.Pasirinkimo mygtukai

Kurdami mygtukų grupę paprastai naudojame tam skirtą komponentą – „pasirinkimo mygtukų grupė“ (angl. *RadioGroup*, piktograma ).

Sukurta grupė pradžioje būna tuščia, su neužpildytomis pozicijomis. Formos projektavimo režime jos užpildomos pasinaudojus savybe **Items**, kurio parametras – tekstas.

Sąraše esančių įrašų kiekį parodo parametras **Count**. Parašę **RadioGroup1.Items.Count** sužinotume, keik sąraše yra įrašų.

Tačiau kartais prireikia sąrašą papildyti programiškai, t. y. programos vykdymo metu. Tam naudojama procedūra **Add(s)**, kurios parametras *s* yra eilutės tipo. Pvz.:

RadioGroup1.Items.Add('dar vienas mygtukas')

Taip pat yra galimybė duomenis į sąrašus nuskaityti iš bylos:

RadioGroup1.Items.LoadFromFile(’kelias iki bylos**’);**

Taip pat programiškai galime ir trinti įrašus. Tam naudojama procedūrą **Delete(i)**, kurios parametras i yra *integer* tipo. Tačiau trinant įrašą reikia žinoti jo numerį sąraše. Tam naudojama funkcija **IndexOf(s),** kurios parametras s yra *string* tipo kintamasis, o rezultatas – *integer*.

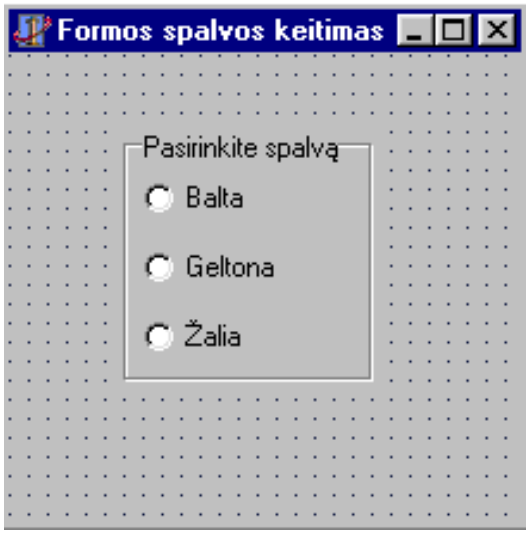
Parašius sakinį **RadioGroup1.Items.Clear** panaikinti visi mygtukai.

Jei žinodami i–tosios eilutės numerį norėtume sužinoti jos tekstą, tai turėtume parašyti: **RadioGroup1.Items[i].**

Programos vykdymo metu sąraše pasirinktos eilutės numerį parodo procedūra **ItemIndex**, kurios rezultatas yra integer tipo kintamasis.

PAVYZDYS.

Sukurkime formą (pav. 9), kurios lange būtų pasirinkimo mygtukų sąrašas, kuriais būtų keičiama formos spalva.



pav. 9

Kadangi veiksmas vyksta sąraše, tai įvykis rašomas taip pat sąrašui:

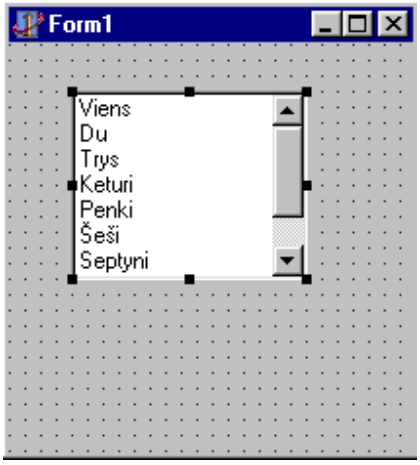
```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
   case RadioGroup1.Itemindex of
     0 : Form1.color := clwhite;
     1 : Form1.color := clyellow;
     2 : Form1.color := clgreen;
   end;
end;
```

PRAKTINĖ UŽDUOTIS

- Sudarykite programą, kurios lange būtų trys tekstinės eilutės: dviejose įrašomi skaičiai, o trečioje parašomas veiksmo rezultatas. Veiksmą (sudėti, atimtis, daugyba, dalyba) parenkame pasirinkę atitinkamą pasirinkimo mygtuką.

14.Sąrašas

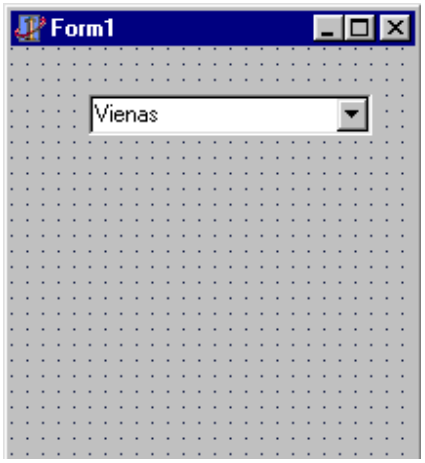
Sąrašas (angl. **ListBox**, pav. 10 taip pat skirtas pasirinkti vieną iš sąrašo alternatyvų



pav. 10

Slinkties juosta šalia sąrašo atsiranda automatiškai, jei visas sąrašas nesutelpa į išskirtą vietą.

Sąrašo atmaina – išsiskleidžiantis sąrašas (angl. **ComboBox**, pav. 11).



pav. 11

Sukurtas sąrašas pradžioje būna tuščias, su neužpildytomis pozicijomis. Programos redagavimo metu jos užpildomos pasinaudojus Objektų Inspektoriaus metodu *Items*, kurio parametras – tekstas.

Sąraše esančių įrašų kiekį parodo parametras **Count.** Parašę **ListBox1.Items.Count** (**ComboBox1.Items.Count**) sužinotume, keik sąraše yra įrašų.

Tačiau kartais prireikia sąrašą papildyti programiškai, t. y. programos vykdymo metu. Tam naudojama procedūra **Add(s)**, kurios parametras *s* yra eilutės tipo. Taip pat yra galimybė duomenis į sąrašus nuskaityti iš bylos:

ListBox1.Items.LoadFromFile('kelias iki bylos**');** (**ComboBox1.Items.LoadFromFile('**kelias iki bylos**')).**

Taip pat programiškai galime ir trinti įrašus. Tam naudojama procedūrą **Delete(i),** kurios parametras i yra *integer* tipo. Tačiau trinant įrašą reikia žinoti jo numerį sąraše. Tam naudojama funkcija **IndexOf(s),** kurios parametras s yra *string* tipo kintamasis, o rezultatas – *integer*. Parašius sakinį **ListBbox1.Clear** (**ComboBox1.Clear**) būtų panaikinti visi sąrašo įrašai.

Kartais yra naudinga, kai įrašai yra išdėstyti abėcėlės tvarka. Tam savybei *Sorted* turine suteikti *True* reikšmę.

Jei žinodami i–tosios eilutės numerį norėtume sužinoti jos tekstą, tai turėtume parašyti: **ListBox1.Items[i]** (**ComboBox1Items[i]**).

Programos vykdymo metu sąraše pasirinktos eilutė numerį parodo procedūra *ItemIndex*, kurios rezultatas yra *integer* tipo kintamasis.

PRAKTINĖ UŽDUOTIS

- Praeito skyrelio pavyzdį pakeiskite taip, kad formos spalvą būtų galima pasirinkti iš išsiskleidžiančio sąrašo.

15.Valdymo meniu

Delphi sistemoje galima naudoti dviejų tipų meniu: pagrindinį meniu (angl. *MainMenu)* ir vietinį meniu (angl. *PopupMenu)*. Pagrindinis meniu – tai meniu, kuris visada būna po formos pavadinimu, o vietinis meniu iškviečiamas ant objekto paspaudus dešinįjį pelės mygtuką. Kadangi programoje gali būti daug objektų, tai vietinių meniu gali būti taip pat daug. Kiekvienas objektas turi savybę *PopupMenu*, kuria galima nurodyti, kuris iš sukurtų vietinių meniu yra skirtas objektui.

Kai formoje jau yra įkeltas meniu objektas  , dukart spragtelėjus ant jo, ekrane atveriamas pagrindinio meniu projektavimui skirtas langas (pav. 12).

pav. 12

Punktyrinis stačiakampis leidžia toje vietoje įvesti naują vardą.

Naudingas meniu komponentus yra skiriamoji juosta – horizontali, nereaguojanti į jokius įvykius linija, kuria atskiriamos atskiros komandų grupės. Sukuriama vietoje *Caption* įrašius brūkšnelį „-“.

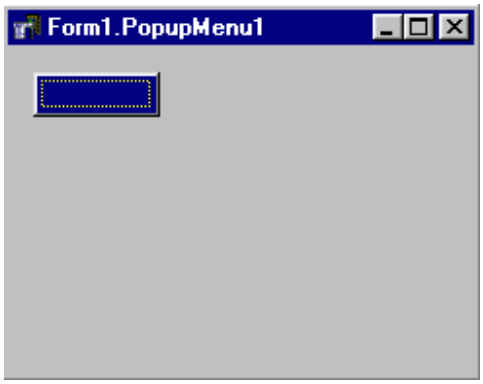
Taip pat dažnai sutinkama mygtukų kombinacija (angl. *Shortcut*), esanti prie meniu punkto (pavyzdžiui, *New*... *Ctrl+N*), leidžianti pagreitinti darbą. Mygtukų kombinaciją galima pasirinkti iš sąrašo.

Dažnai sutinkame meniu, kuriame naudojamas dar ir meniu papunktis, kurį žymi rodyklė į dešinę (pav. 13). Norint meniu punktui sukurti papunktį, reikia esant ant to meniu paspausti mygtukų kombinaciją *Ctrl* + ➔.

pav. 13

Norint parašyti meniu punktui procedūrą, reikia ant to punkto dukart spragtelėti kairiuoju pelės klavišu.

Vietinis meniu kuriamas analogiškai - kai formoje jau yra įkeltas vietinio meniu objektas  , dukart spragtelėjus ant jo, ekrane atveriamas vietinio meniu projektavimui skirtas langas (pav. 14).



pav. 14

PRAKTINĖ UŽDUOTIS.

- Sukurkite programą, kurios lange būtų parašytas sakinys. Programa turi turėti pagrindinį meniu iš dviejų dalių: „Forma“ ir „Tekstas“. Pasirinkus „Forma“, turi atsiverti komandų sąrašas: „Spalva“, „Dydis“, „Išeiti“. Pasirinkus „Tekstas“, turi atsiverti komandų sąrašas: „Spalva“, „Dydis“. Abiejose dalyse (tiek „Forma“, tiek „Tekstas“) pasirinkus „Spalva“, turi atsiverti papunktis su galimų pasirinkti spalvų pavadinimais, o pasirinkus „Dydis“, turi atsiverti papunktis, kuriame būtų galima pasirinkti: didinti ar mažinti norime objektą.
- Papildykite programą vietiniais meniu, kurių vienas keistų teksto dydį, o antras – formos.

16. Daugialangė sąsaja

Dauguma programų turi pagrindinį sąsajos langą ir vieną ar keletą, net kelioliką pagalbinių langų, kuriuose pateikiami skaičiavimų rezultatai arba informaciniai duomenys. Tokiu atveju kiekvienam langui projektuojama atskira forma, kiekvienai formai rašomas atskiras modulis. Pagalbiniai moduliai prijungiami prie pagrindinio naudojant komandą **uses.** Pagalbiniai langai iškviečiami procedūra **ShowModal** arba **Show**. Pvz. Form2.ShowModal. Jei norime, kad nebūtų galima grįžti į pagrindinį langą neuždarius pagalbinio, naudojame **ShowModal**, priešingu atveju - **Show**.

PRAKTINĖ UŽDUOTIS.

- Papildykite praeito skyrelio meniu dar vienu punktu – „Apie autorių“. Jį pasirinkus, turi atsidaryti langas, kuriame būtų informacija apie jus.

Literatūra

1. Vidžiūnas A. ir kt. Delphi 5 programavimo pavyzdžiai. – Kaunas: Smaltija, 2000.
2. Petkevičius T. Programavimo sistema Delphi.