Vilniaus Gedimino technikos universitetas

Algirdas Sokas

Programavimas VBA kalba

Mokomoji knyga

Vilnius „Technika“ 2005

A. Sokas. Programavimas VBA kalba. Mokomoji knyga. Vilnius: Technika, 2005. 53p.

Leidinyje pateikiamos *Visual Basic* programavimo kalbos varianto VBA (*Visual Basic for Application*) programavimo galimybės, supažindinama su redaktoriaus aplinka ir objektais, pateikiami kintamųjų ir konstantų taikymo principai, analizuojamos operatorių savybės, rašomos pirmosios procedūros, nagrinėjamos valdymo struktūros ir galimybės sudaryti vartotojo funkcijas, pateikiamos vidinės programavimo kalbos funkcijos. Mokomojoje knygoje supažindiname su klasės objektų programavimu, objektų savybių ir metodų kūrimo būdais ir pavyzdžiais, galimybėmis programiniu būdu perskaityti ir užrašyti informaciją teksto rinkmenoje.

Knyga skirta Inžinerinės grafikos katedros magistrantams ir visiems, kas nori išmokti programuoti VBA kalba.

Leidinį rekomendavo Fundamentinių mokslų fakulteto studijų komitetas

Recenzavo doc. dr. P. Gerdžiūnas ir doc. dr. R. Kutas

VGTU leidyklos „Technika“ 755 mokomosios metodinės literatūros knyga

ISBN 9986-05-834-1

Turinys

1. Įvadas

Pirmoji programavimo kalbos *BASIC* versija buvo skirta pradedantiesiems mokytis programuoti. Ją 1963 metais Dartmaunto (*Dartmount*) koledže sukūrė amerikiečiai Džonas Kemenis (*John Kemeny*) ir Tomas Kurtzas (*Thomas Kurtz*). Žodžio *BASIC* pirmosios raidės reiškia – pradedantiesiems daugiatikslis simbolinių instrukcijų kodas (*Beginners Allpurpose Symbolic Instruction Code*). Kalba buvo sukurta programavimo kalboms mokyti, pagrindams suvokti ir paprastoms programoms rašyti.

1975 m. jaunas programuotojas Paulas Alenas (*Paula Allen*) ir Harvardo universiteto pirmakursis Bilas Gatesas (*Bill Gates*) parengė „*Altair*“ kompiuteriui pirmąją *BASIC* sistemą. Vėliai šie jaunuoliai įkūrė garsiąją „*Microsoft*“ kompaniją.

1991 m. „Microsoft“ kompanija sukūrė *Visual Basic* (VB) programavimo kalbą, skirtą dirbti sistemos *Windows* aplinkoje ir naudoti jos išteklius.

VB – tai jau objektinio programavimo kalba. Kalba valdo objektus, kuriems atliekami įvairūs veiksmai. Pasikeitė programavimo aplinka – atsirado galimybės stebėti projekto vykdymą, projektuoti priedus. Dabar visas programos kodas paskirstytas į procedūras (paprogrames), kurios redaguojamos ir iškviečiamos atskirai.

VBA (*Visual Basic for Application*) yra VB kalbos variantas, kuris naudojamas *Microsoft Office* programose (*Word, Excel, PowerPoint, Outlook, Access*) ir „Autodesk“ grafinėje sistemoje *AutoCAD*.

Galimybė programuoti grafinėje aplinkoje labai sudomino ir privertė nuodugniau studijuoti VBA programavimo kalba, o ši mokymo knyga yra pirmasis žingsnis į grafikos programavimą *AutoCAD* aplinkoje.

Skyriuje „Redaktorius“ pateikiamas VBA redaktorius, esantis *Microsoft Office* programose. Jame supažindinama su projektų valdymu vadovo lange, analizuojamos grafinio redaktoriaus objektų, savybių, bibliotekų galimybės, nagrinėjamos teksto redaktoriaus savybės, dialogo langai ir priedai. Trumpai pateikiamas standartinis įrankių meniu.

Skyriuje „Kintamieji ir konstantos“ susipažinsime su programavimo kalbos *Visual Basic* kintamųjų ir konstantų įvardijimu, apribojimais, duomenų tipais, matomumu ir deklaravimu.

Skyriuje „Operatoriai“ pateikiami VB programavimo kalbos priskyrimo, matematiniai, lyginimo ir loginiai operatoriai ir jų prioritetų seka.

Skyriuje „Procedūros“ supažindiname su procedūrų rašymu ir jų matomumo naudojimu.

Skyriuje „Valdymo struktūros“ pateikiamos sąlyginės, besąlyginės ir ciklinės valdymo struktūros su blokinėmis schemomis, sintaksės išraiškomis ir pavyzdžiais.

Skyriuje „Funkcijos“ aptariamos vidinės programavimo kalbos funkcijos ir galimybės sudaryti ir taikyti vartotojui funkcijas.

Skyriuje „Klasės“ supažindiname su klasės objektų programavimu, objektų savybių ir metodų kūrimo būdais ir pavyzdžiais, pateikiama objekto *kolekcija* supratimas ir metodai.

Skyriuje „Rinkmenos“ nagrinėsime, kaip galima programiniu būdu perskaityti ir užrašyti teksto rinkmenoje informaciją.

Mokomosios knygos pabaigoje pateiktas literatūros sąrašas.

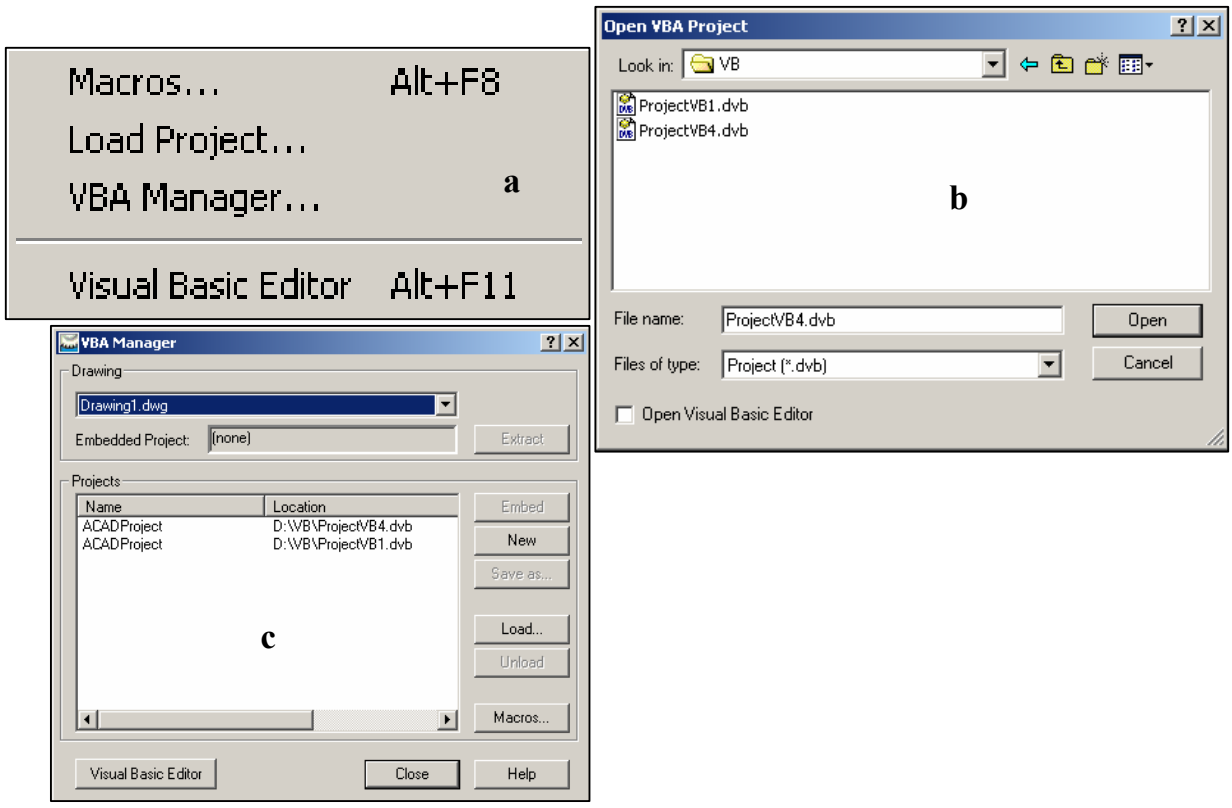
Teksto poskyriai, komandos, funkcijos, paprogramių pradžia ir pabaiga rašomos ryškesniu šriftu, meniu langų pavadinimai rašomi kabutėse, mygtukai – laužtiniuose skliausteliuose. Tekste lietuviško termino angliškas atitikmuo pateikiamas pasviruoju šriftu skliausteliuose, programos tekstas ir angliški pavadinimai irgi rašomi pasviruoju šriftu.

Mokomoji knyga skirta Inžinerinės grafikos katedros magistrantams ir visiems, kas nori išmokti programuoti VBA kalba.

2. Redaktorius

VBA (*Visual Basic for Application*) redaktorius „*Microsoft*“ programose (*Word, Excel, PowerPoint, Outlook, Access*) ir „*Autodesk*“ grafinėje sistemoje *AutoCAD* paleidžiamas per meniu **Tools/Macro** suaktyvinus komandą **Visual Basic Editor** (1 a pav.).

Kviečiant anksčiau sukurtą projektą, komanda **Load Project** aktyvinamas meniu langas „Open VBA Project“ (1 b pav.) arba komanda **VBA Manager** aktyvinamas meniu langas „VBA Manager“ (1 c pav.) ir pasirinkus iškviečiamas atitinkamai su mygtuku [Open] arba [Load] .

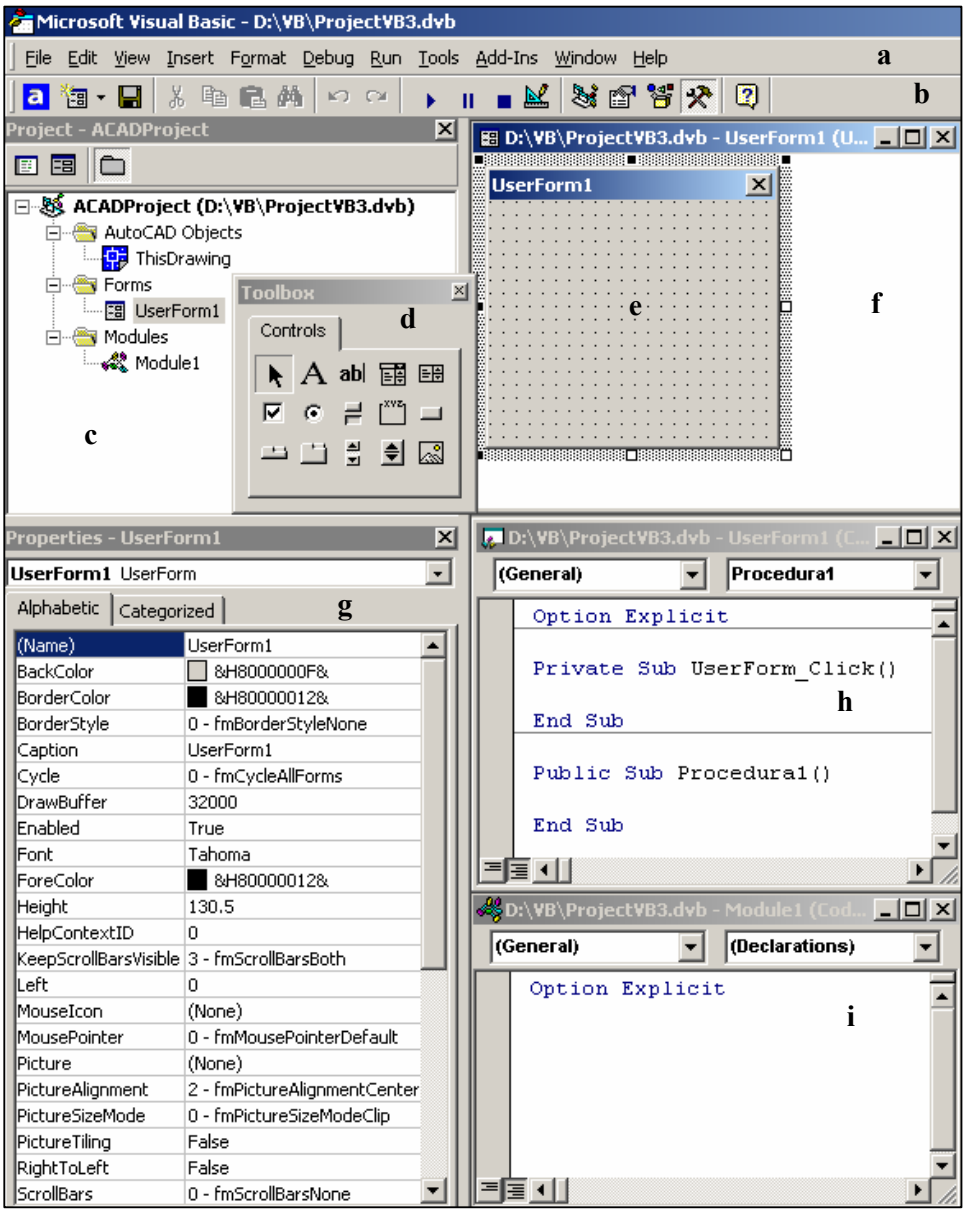


1 pav. Projektų iškvietimo ir kūrimo meniu ir dialogo langai:
a – meniu, b – projektų iškvietimo langas, c – projektų valdymo langas

Iškviečiant projektą, visi projektą sudarantys elementai (dialogo langai, moduliai, klasės ir t. t.) bei redaktorius pakraunami automatiškai.

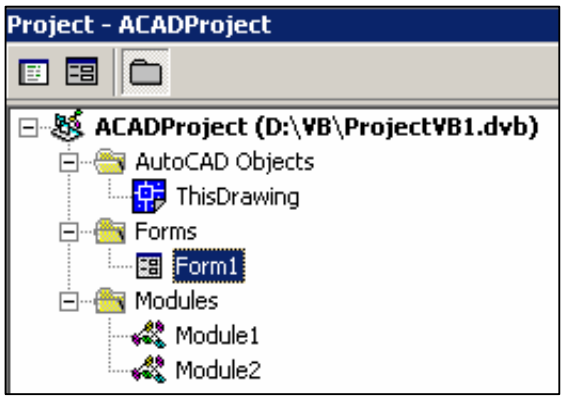
VBA projektams kurti skiriami redagavimo laukai. Redaguojant dialogo langą, naudojamas grafinis redagavimo laukas, redaguojant projekto kodus – teksto redagavimo laukas. Kiekvienam dialogo langui, projektui, moduliui skiriamas tam tikras redagavimo laukas (2 pav.).

Projekto vadovo langas (*Project explorer*). Matyti visus projekto elementų laukus ir redaguoti juose nėra paranku, todėl geriau redaguojamam laukui turėti kuo daugiau erdvės. Lengviau ir greičiau pereiti iš vieno lango į kitą reikia projekto vadovo (2 c pav.), kadangi jame yra visų, esančių projekte ar projektų grupėje, elementų sąrašas. Pereiti į norimą elemento redaktoriaus lauką pasirenkamas elementas iš projekto vadovo sąrašo ir spaudžiamas projekto vadovo mygtukas [View Object] pereiti į grafinio redaktoriaus lauką arba [View Code] pereiti į teksto redaktoriaus lauką. Pereiti į aktyvaus projekto elemento kodų ar dialogo lango redaktoriaus lauką galima ir iš pagrindinio meniu – atitinkamai **View / Object** arba **View / Code**.



2 pav. VBA redaktorius: a – pagrindinis meniu, b – standartinis įrankių meniu, c – projekto vadovas, d – objektų meniu, e – dialogo lango šablonas, f – grafinio redaktoriaus laukas, g – objektų savybių laukas, h – teksto redaktoriaus laukas, i – teksto redaktoriaus laukas moduliui

Kai projekte yra daugiau dialogo langų, modulių ar kitų projektų, pereiti iš vieno projekto, dialogo lango ar modulio redaktoriaus lauko į kitą galima projekto vadovo lange (3 pav.). Tam reikia iš projekto vadovo lange pateikiamo konkretaus projekto elementų sąrašo išsirinkti reikiamą elementą bei redaktoriaus lauko tipą. Primenu, kad grafinio redaktoriaus lauke redaguojami dialogo langai (formos), tai yra keičiamas dialogo lango dizainas, o teksto redaktoriaus lauke yra rašomi projekto kodai. Tada, kai nuspręsite, kurį redaktorių naudoti, spauskite reikiamą mygtuką. Tie mygtukai yra projekto vadovo viršutinėje dalyje. Reikalingą elementą galima tiesiogiai su pele pasirinkti projekto vadovo lange.



3 pav. Projekto vadovo langas

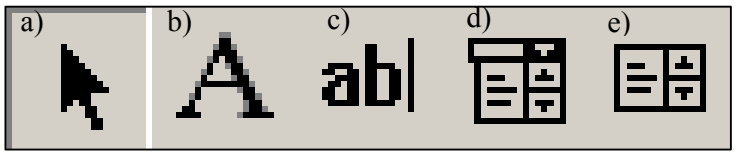
Grafinio redaktoriaus laukas (*Object*). Grafinio redaktoriaus laukas iškviečiamas iš pagrindinio programos meniu **View/Object** arba paspaudus projekto vadovo dialogo lango mygtuką [View Object]. Grafinis redaktorius skirtas dialogo langų objektams redaguoti ir dialogo langų dizainui. Iškviesti reikiamus objektus į dialogo langą nesunku, bet tuos objektus reikia komponuoti, kad dialogo lango vaizdas būtų suprantamas, stilingas, racionalus. Tai priklauso nuo programuotojo meninių, intelektinių gebėjimų. Visi projektai skiriami vartotojui, todėl būtina skirti šiek tiek laiko ir dialogo langams apipavidalinti.

Objektą apibūdina duomenys vadinami savybėmis (*properties*). Pavyzdžiui, grafinių objektų savybės apibrėžia jų dydį, padėtį, spalvas, užrašus, matomumą ir panašiai. Procedūros, galinčios apdoroti su objektu ir jo savybėmis susijusią informaciją, vadinamos metodais (*methods*).

Pagrindiniai VBA grafiniai objektai yra formos ir valdikliai. Formomis vadinami langai, kuriuose yra valdikliai: mygtukai, reguliatoriai, meniu. **Forma** (*form*) yra pagrindinis objektas, nuo kurio pradedama kurti programa. Programa gali turėti keletą formų, kurių matomumas keičiasi dirbant. Kiekviena forma tai projektavimo langas, kuris turi du režimus: objektų grafinio modeliavimo (*object*) ir programos kodo rašymo (*code*).

Objektų grafinio modeliavimo lange turime formą, kurioje, naudodamiesi objektų meniu „Toolbox“ (2 b pav.), galime modeliuoti programos langą, keisti objektų savybes. Programos kodo rašymo lange turime visų mygtukų ir kitų objektų įvykių tuščias procedūras, kurias galime išplėsti reikalingais veiksmais.

Objektų meniu paprasčiausias grafinis objektas yra **užrašas** (*Label*), kurios piktograma pateikta 4 b pav. Ji naudojama pavadinimams ir užrašams rašyti pačioje formoje.

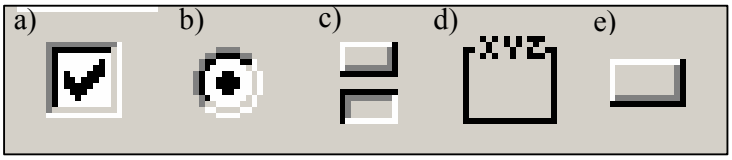


4 pav. Objektų piktogramos: a – žymeklis (*Pointer*), b – užrašas (*Label*),
c – teksto eilutė (*TextBox*), d – pasirinktinė įvestis (*ComboBox*), e – sąrašas (*ListBox*)

Teksto eilutė (*TextBox*) – informacijos įvedimo arba išvedimo eilutė, kuri leidžia vartotojui pateikti informaciją programai dirbant arba pačiai programai atspausdinti nurodytą informaciją (4 c pav.).

Pasirinktinė įvestis (*ComboBox*) – informacijos pasirinkimo ir įvedimo laukas, kuriame galima pasirinkti vieną elementą iš sąrašo arba įvesti kitokią informaciją (4 d pav.).

Sąrašas (*ListBox*) – informacijos pasirinkimo laukas, kuriame galima pasirinkti vieną arba kelis elementus (4 e pav.).



5. pav. Objektų piktogramos: a – jungiklis (*CheckBox*), b – pasirinkimo mygtukai (*OptionButton*), c – jungtukas (*TogleButton*), d – rėmas (*Frame*), e – mygtukas (*CommandButton*)

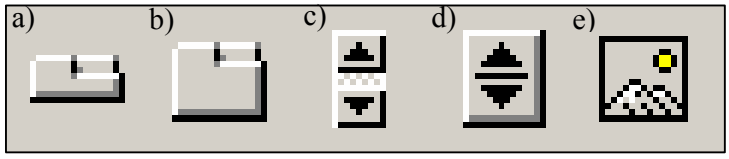
Jungiklis (*CheckBox*) skirti kokiam nors režimui įjungti arba išjungti, būklei keisti ir panašiai (5 a pav.). Jungikliai veikia nepriklausydami vienas nuo kito. Jungiklio būklę rodo savybės *Value* reikšmė. Jei jungiklis įjungtas, ji lygi 1, jei išjungtas – 0.

Pasirinkimo mygtukai (*OptionButton*) dažniausiai yra sugrupuoti ir veikiant programai galima pasirinkti tiktai vieną, nes jie veikia išvien (5 b pav.). Pasirinkimo mygtuko būklę rodo savybės *Value* reikšmė. Jei mygtukas įjungtas, ji lygi tiesos (*True*) reikšmei, jei išjungtas – melo (*False*) reikšmei. Iš visos grupės tik vieno mygtuko savybės *Value* reikšmė būna *True*, kitų reikšmės automatiškai tampa lygios *False*.

Jungtukas (*ToggleButton*) skirti vienam režimui įjungti arba išjungti (5 c pav.). Jungtuko būklę rodo savybės **Value** reikšmė. Jei jungtukas įjungtas, ji lygi tiesos (*True*) reikšmei, jei išjungtas – melo (*False*) reikšmei.

Rėmas (*Frame*) – skirtas formoje vizualiai sugrupuoti valdymo objektus, tokius kaip pasirinkimo mygtukus ar jungiklius (5 d pav.).

Mygtukas (*CommandButton*) – skirtas programai arba tam tikroms komandoms įjungti (5 e pav.).



6 pav. Objektų piktogramos: a – sudėtinė kortelė (*TabStrip*), b – sudėtinis puslapis (*MultiPage*), c – šliaužiklis (*ScrollBar*), d – skaitliukas (*SpinButton*), e – paveikslėlis (*Image*)

Sudėtinė kortelė (*TabStrip*) – skirta informacijos kolekcijai pateikti (6 a pav.).

Sudėtinis puslapis (*MultiPage*) – skirta eilės ekranam pateikti (6 b pav.).

Šliaužiklis (*ScrollBar*) – pateikia vieną ne neigiamąją ir sveikąją skaitmeninę reikšmę iš nurodytos reikšmių srities (6 c pav.).

Skaitliukas (*SpinButton*) – pateikia po vieną sveiką skaičių didėjančia seka arba gautą skaičių analogiškai galime mažinti (6 d pav.).

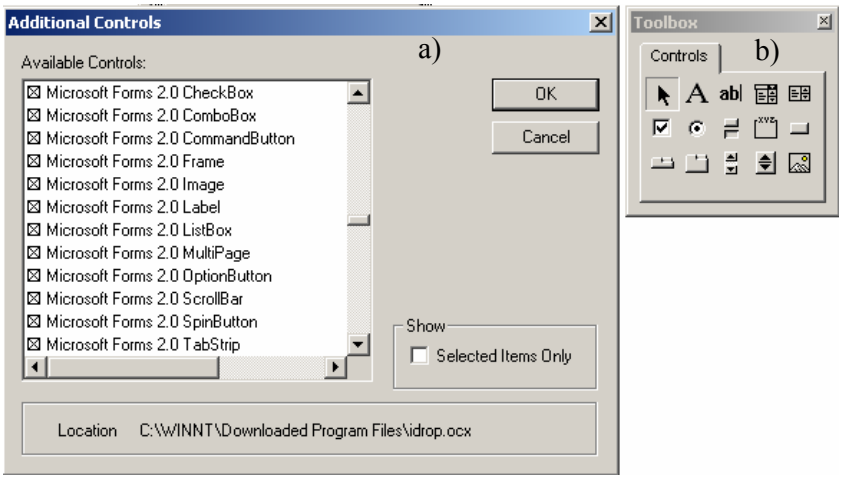
Paveikslėlio laukas (*Image*) – skirtas įvairių tipų grafinėms rinkmenoms pavaizduoti (6 e pav.).

Atsižvelgiant į programavimo stadiją, bus „pasiekiamas“ ar „nepasiekiamas“ kuris nors įrankių meniu mygtukas, mygtuku valdoma komanda. Stadijomis reikėtų vadinti dialogo lango kūrimą, jo redagavimą, programos kompiliavimą. Atitikmenis mygtukams galima surasti ir pagrindiniame meniu.

Pageidaujant galima susikurti asmeninį įrankių meniu, tik pakeisti mygtukų komandų ar sukurti naujų komandų – negalima.

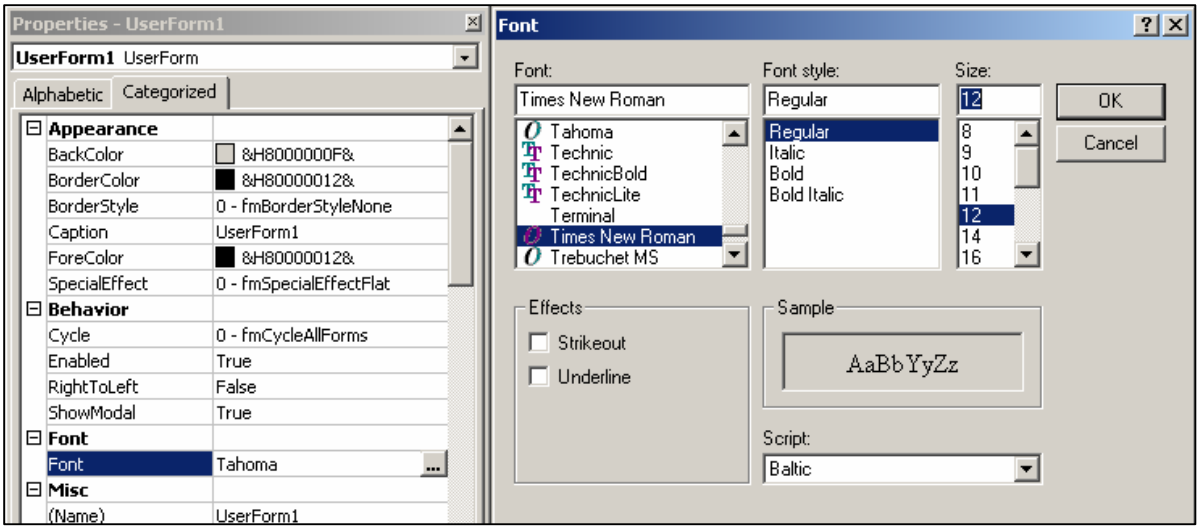
Objektų pasirinkimo įrankių dėžutė (*Toolbox*). Kai nepakanka standartinių objektų, galima atsikelti juos iš objektų rinkinių bibliotekos (7 pav.). Tam reikia aktyvinti meniu „Tools“ ir pasirinkti komandą **Additional Controls**, dialogo lange išsirinkti norimus objektus ir nuspausti mygtuką [OK]. Įkeltieji objektai savaime atsiras objektų lauke (*Toolbox*).

Objektų priskyrimo dialogo langui (formai) grafinio redaktoriaus lauke reikia pažymėti objektą ir dialogo lango ribose paspaudus kairįjį pelės mygtuką nurodyti objekto dydžius.



7 pav. Įrankių dėžutė: a – meniu, b – įrankiai

Objekto savybių langas (*Properties*). *Microsoft Visual Basic* yra objektiškai orientuota programavimo kalba, todėl kiekvienas dialogo lango (formos) elementas ar deklaruotas projekto kintamasis yra priskiriamas objektui arba gali būti jam priskirtas. Objektu laikomas ir dialogo langas. Kiekvienam dialogo lango elementui skiriamas tam tikras savybių rinkinys. Šias savybes peržiūrėti ar pakeisti kokios nors savybės reikšmę galima objektų savybių lange (8 pav.).



8 pav. Objekto savybių langas su aktyviu šriftų dialogo langu

Keičiant kurią nors kurio nors dialogo lango elemento savybę lange „Properties“, pirmiausia reikia aktyvinti tą objektą. Kiekvienam elementui skiriamas jį aprašantis savybių sąrašas, todėl aktyvinimui reikia aktyvaus elemento savybes pateikti lange „Properties“. Skirtingiems elementams šis sąrašas skirtingas. Aktyvinti norimą elementą galima pasirinkti jį žymekliu grafinio redaktoriaus lauke arba parenkant elemento vardą iš lango „Properties“ išskleidžiamo elementų vardų sąrašo.

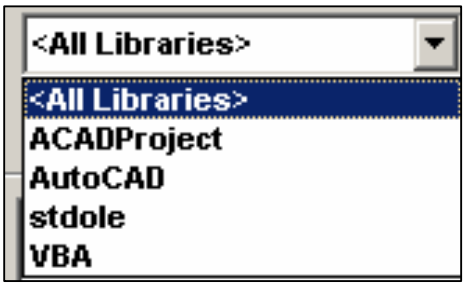
Keičiant savybę, reikia ją pažymėti norimoje grafoje ir jos parametrą įvesti arba pasirinkti iš siūlomų reikšmių.

1 lentelė. Kai kurios dialogo langui priklausančių elementų savybės

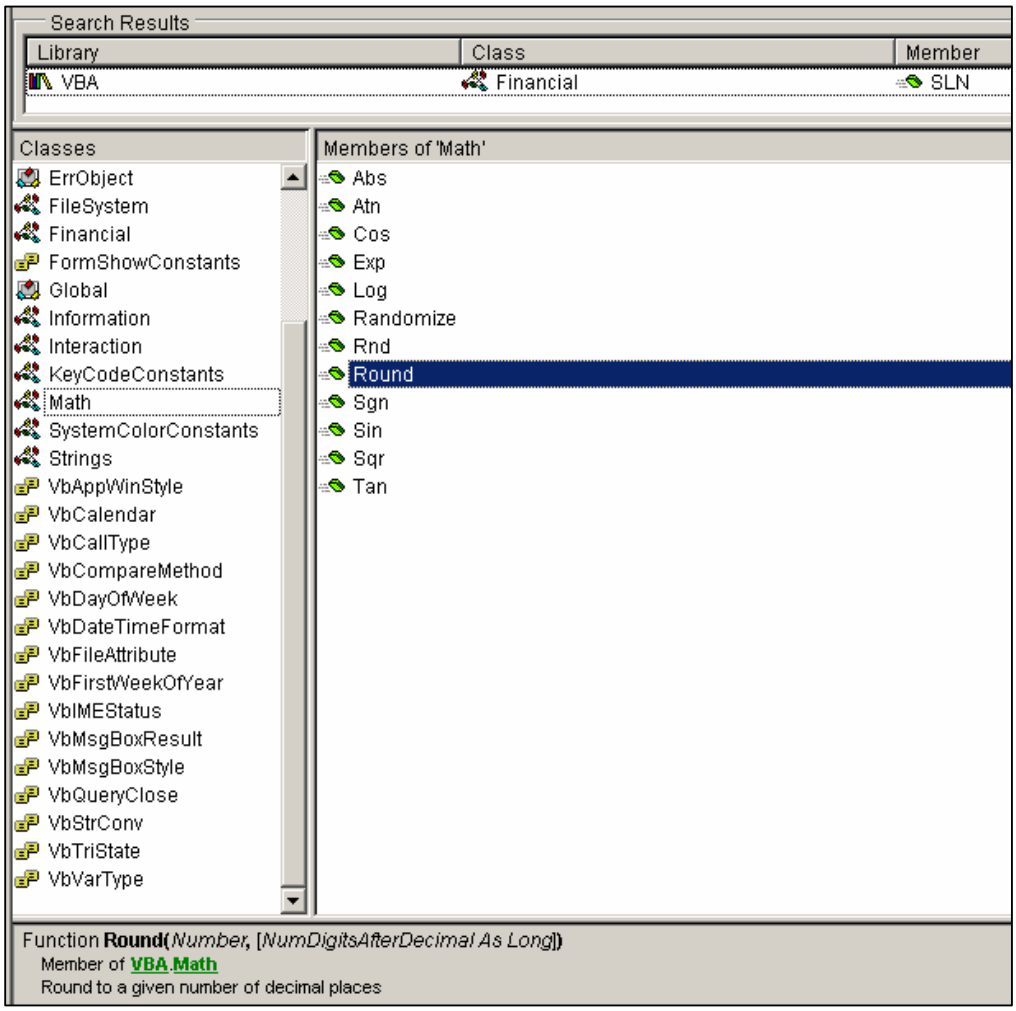
Savybė	Pvyzdys	Apibūdinimas
(Name)	*„Data1“*	Vardas, kuriuo kreipiamasi į objektą
BackColor	*&H8000000F&*	Fono spalva
BorderColor	*&H80000012&*	Objekto rėminimo spalva
BorderStyle	*2*	Objekto rėminimas
Caption	*„Langas“*	Užrašas ant objekto
Connect	*Access*	Prisijungimo prie duomenų bazės tipas
DatabaseName	*„c:\db.mdb“*	Duomenų bazės vardas
DataField	*„varžtai“*	Duomenų stulpelis (lentelės stulpelio vardas)
DataSource	*„Data1“*	Duomenų šaltinis (*Data* objekto vardas)
Enabled	*True*	Objekto veiksnumo apribojimas
Font	*MS Sans Serif*	Šriftas
Forecolor	*&H80000012&*	Teksto spalva
Height	*3600*	Objekto aukštis vienetais
Icon	*(Icon)*	Piktograma
Left	*50*	Objekto atstumas iki kairiojo formos krašto
List	*„Labas“*	Įrašų sąrašas
Locked	*False*	Redagavimo galimumas
Picture	*(Bitmap)*	Fonas kaip grafinis vaizdas
RecordSource	*„Detalės“*	Įrašų šaltinis (lentelės vardas)
ScaleHight	*1000*	Formos vertikalusis dalijimas
ScaleWidth	*2000*	Formos horizontalusis dalijimas
SpecialEffect	*1*	Objekto tipai
Tag	*„A001“*	Atpažinimo ženklas
Text	*„Labas“*	Įrašas objekte
TextAlign	*1*	Įrašo lygiavimas lauke
TooltipText	*„Vykdyti“*	Pagalbinis tekstas
Visible	*False*	Objekto matomumas
Width	*4800*	Objekto plotis vienetais
WindowState	*0*	Dialogo lango būsena
WordWrap	*True*	Sakinys rašomas keliose eilutėse
Zoom	*100*	Objektų mastelis formoje

Dialogo lango elemento vieną ar kitą savybę galima keisti ne tik lange „Properties“, bet ir projekto kodų eilutėse (teksto redaktoriaus lauke). Dažniausiai lange „Properties“ nustatomos nekintančios ir pradinės elementų savybės, o kintančios – projekto koduose.

Objektų bibliotekų sąrašas (*Object Browser*). Šis langas skirtas objekto savybių, metodų, funkcijų paprogramių, taikytinų objektui, konstantų, įvykių paieškai. Šio dialogo lango duomenų šaltinis yra bibliotekų rinkiniai, kuriose išvardyti visi bruožai, taikomi objektams, aplinkoms, sistemoms. Visi šie duomenys suskirstyti į bibliotekas, todėl ieškant bruožų galima taikyti konkrečius atvejus ir reikia ieškoti informacijos konkrečiai bibliotekai priklausančioje klasėje.



9 pav. Objektų bibliotekų langas

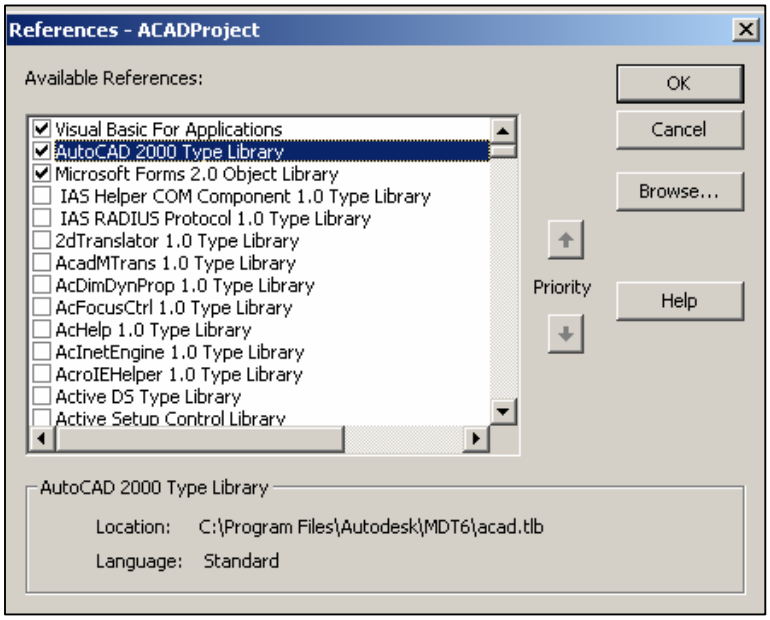


10 pav. Objektų bibliotekos matematikos klasės narių sąrašas

Kad bruožo paieška vyktų tik vienoje išsirinktoje bibliotekoje, reikia tą biblioteką aktyvinti. Aktyvios bibliotekos vardas šviečia objektų bibliotekų sąrašo (*Object Browser*) dialogo lango laukelyje. Bruožui ieškoti visose pakrautose bibliotekose skirtas vardas **All Libraries**.

Į objektų bibliotekų sąrašo turinį automatiškai pakraunamos tik kelios klasės (*VB, VBA, VBRUN, stdole, Project1*), todėl kitų objektų bibliotekos iškviečiamos savarankiškai. Objektų bibliotekas pakrauti galima išsikviečiant dialogo langą „References“ iš pagrindinio programos meniu **Tools/References**… arba iš objektų bibliotekų sąrašo dialogo lango „Object Browser“ savybių meniu **References...** (savybių meniu iškviečiamas paspaudus dešinįjį pelės mygtuką). Atsiradusiame lange išsirenkamos tos bibliotekos, kurių reikia, ir pasirinkimas patvirtinamas mygtuku [OK]. Neradus norimos bibliotekos pavadinimo, galima pamėginti susirasti ją kitur. Bibliotekai pakrauti iš kitur skirtas paieškos mygtukas [Browse…]. Išsaugojus projektą, išsaugojamas pakrautų bibliotekų rinkinys, ir kito redagavimo metu naujai jų pakrauti nereikės.

Visos bibliotekos sudarytos iš klasių ir konstantų sąrašo, kartu kiekviena klasė sudaryta iš klasių rinkinio ir klasių-bruožų ir t. t. Kiekvienas bruožas grindžiamas tik jam skiriama priklausomybe, todėl naudojant bet kurį bruožą projekto koduose būtinas jo tikslus aprašas.



11 pav. Objektų bibliotekų įjungimo langas

Vis dėlto daugiausia dėmesio reikėtų skirti programos kodams rašyti bei sprendžiamiems uždaviniams optimizuoti, nes nuo to priklausys pačios programos darbo našumas. Pirmiausia į dialogo langą sukeliami reikiami objektai, kad būtų išnaudota dialogo langų objektų paprogramių automatinio sukūrimo galimybė. Kiekvienam dialogo lango objekto paprogramiui skiriamas kažkoks įvykis.

Teksto redaktoriaus laukas (*Code*). Teksto redaktoriaus laukas iškviečiamas iš pagrindinio programos meniu **View/Code** arba paspaudus projekto vadovo dialogo lango mygtuką [*View Code*]. Yra dar vienas būdas, kaip pereiti į teksto redaktoriaus lauką. Dusyk spūstelėjus kairiuoju pelės mygtuku (KPM) grafinio redaktoriaus lauke į objektą, pereinama į teksto redaktoriaus lauko projekto kodus, kuriuose bus tą objektą aprašantys įvykių kodai.

Teksto redaktoriaus lauke užrašomos projekto kodų eilutės, kuriomis aprašomi sprendimo, tikslo uždaviniai. Kiekviena programavimo kalba sudaryta iš programavimo terminų, kuriuos sudaro žodžiai, reiškiantys konstantas, metodus, objektus, savybes, funkcijas, įvykius, veiksmus, loginius ir matematinius operatorius ir t. t. Būtent šių terminų sakiniais užpildomos kodų eilutės. Kad sakiniai nebūtų chaotiškai išmėtyti, jie grupuojami į paprogramius (*subroutines*), į atskiras funkcijas (*functions*), į kintamųjų deklaravimą (*Option Explicit*).

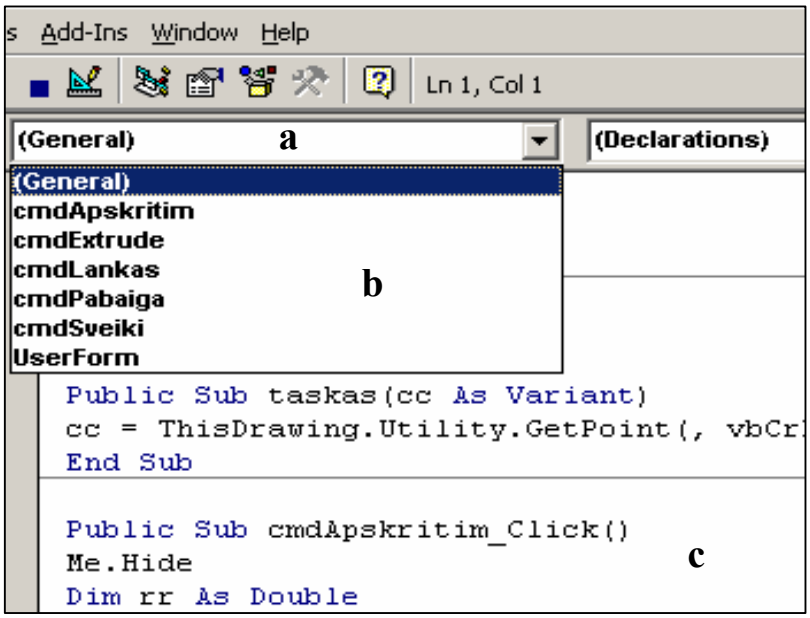
Paprogramiai ir funkcijos automatiškai atskiriami begaliniu horizontaliu brūkšniu. Paprogramio pradžia pažymima žodžiu *Sub*, o pabaiga – *End Sub*. Funkcijos pradžia žymima žodžiu *Function*, o pabaiga – *End Function.* Tarp žodžių *Sub* ar *Function* ir *End Sub* ar *End Function* užrašomi kiti programos kodai.

Kintamieji dažniausiai deklaruojami teksto redaktoriaus lauko viršuje tiek projekte, tiek modulyje, tiek ir klasės modulyje. Tokie kintamieji paprastai yra visiško (*global*) naudojimo. Kintamuosius galima deklaruoti ir paprogramiuose bei funkcijose, tik jų naudojimas gana suvaržytas.

Teksto redaktoriaus lauko skaidymas į deklaravimus, paprogramius ir funkcijas pažymint pradžios ir pabaigos ribas, leidžia lengviau orientuotis lauko erdvėje, ypač kai projekto kodų ne vienas šimtas eilučių. Ieškoti po tūkstančio eilučių lauką norimo kodų sakinio per sunku, todėl yra priemonių pereiti į norimą paprogramį ar funkciją.

Kiekvienam projekto dialogo lango objektui priklauso bent vienas paprogramis. Išsirinkus objektą iš sąrašo, teksto redaktoriaus lauko žymeklis pereis į išsirinktojo objekto paprogramį (12 pav.). Jei vienam objektui priklauso keli paprogramiai, tai žymeklis pereis tik į vieną (abėcėlės tvarka). Kai dialogo lango objekto paprogramio pradžios ir pabaigos kodų nėra – jie bus sukurti automatiškai, jų surinkti pačiam nereikės.

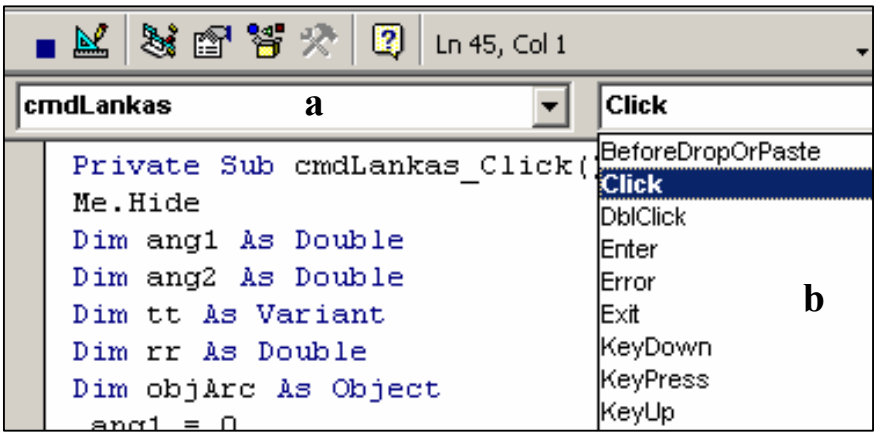
Teksto redaktoriaus lauko žymekliui perkelti į lauko pradžią reikia pasirinkti objektų sąraše eilutę (*General*) (12 pav.).



12 pav. Paieška: a – nuoroda, kad dabar aktyvus šis dialogo lango objekto paprogramis, b – visų projekte esančių dialogo langų objektų vardinis sąrašas, c – teksto redaktoriaus lauko projekto kodai

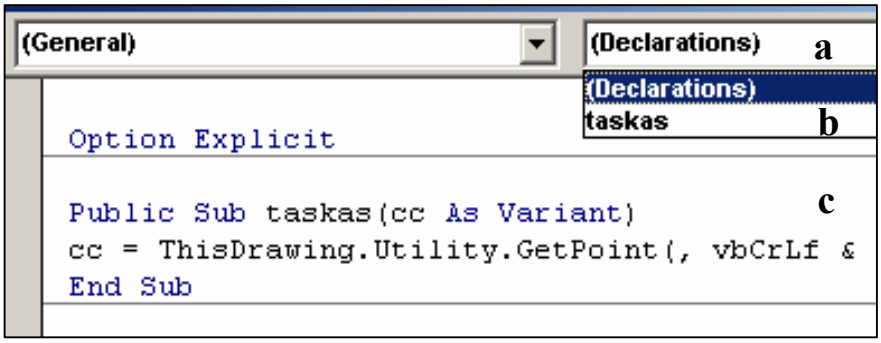
Kiekvienas dialogo lango objekto paprogramis atpažįstamas pagal objekto vardą. Kaip paprogramiai atpažįstami, kai tam pačiam dialogo lango objektui priklauso (projekto koduose naudojami) keli paprogramiai? Paprogramiai įvardijami naudojant dialogo lango objekto vardą ir įvykį. **Įvykis** – projekto tėkmės perdavimo būdas į dialogo lango objekto paprogramį (13 pav.). Įvykių yra įvairių – paspaudimas, pelės žymeklis virš objekto, procedūros iškvietimas, keitimas ir t. t. Skirtingam dialogo lango objektui yra pavaldus atskiras įvykių rinkinys. Kai kurie objektai turi tuos pačius įvykius.

Tėkmės perdavimo į dialogo lango objektų paprogramius būdai (įvykiai) yra tipiniai kiekvienam objektui. VBA automatiškai priskiria dialogo lango objektui įvykį, kuris yra nustatytas kaip dažniausiai naudojamas to objekto (tipiškiausias įvykis). Papildyti kitu įvykiu dialogo lango objektą vertėtų tik norint praplėsti projekto tėkmės lankstumą valdant tą objektą.



13 pav. Paieška: a – dialogo lango objektas, kuriam parenkamas įvykis, b – dialogo lango objektui „cmbLankas“ priskirtas įvykis – paspausti (*Click*)

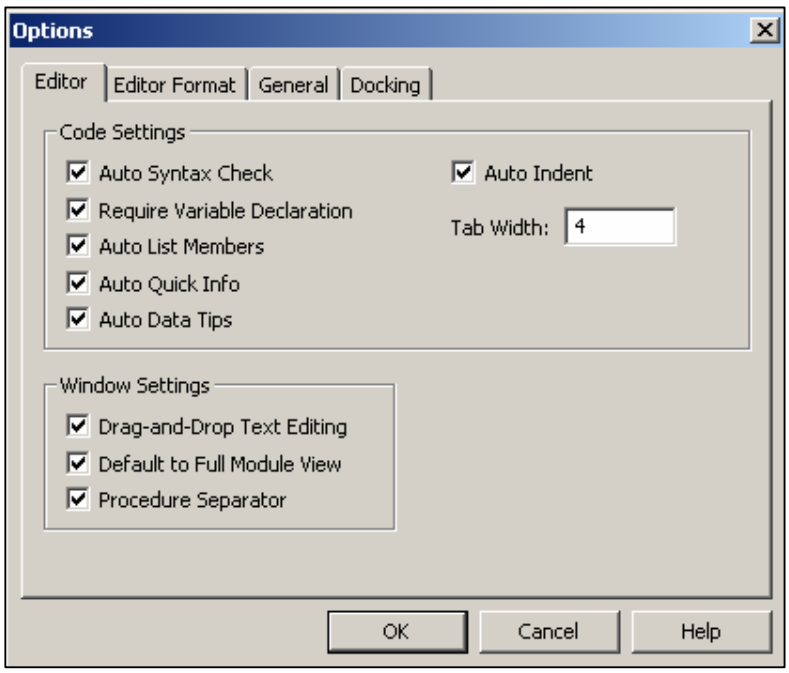
Nepriklausančių dialogo lango objektams paprogramių ir funkcijų paieška teksto redaktoriaus lauke galima, kai neaktyvus joks dialogo lango objektas (aktyvus *General*). Priešingai – deklaruotų paprogramių ir funkcijų sąrašas bus nepasiekiamas. Teksto redaktoriaus lauko žymeklio perkėlimas į lauko pradžią įvyks išsirinkus deklaruojamų paprogramių ir funkcijų vardų sąraše žodį *Declarations*. Žymeklis bus perkeltas į paprogramio pradžią išsirinkus jo deklaruojamą pavadinimą (14 pav.).



14 pav. Paieška: a – perėjimas į teksto redaktoriaus lauko pradžią, b – deklaruotų paprogramių ir funkcijų vardų sąrašas, c – paprogramis „taškas“ ir į paprogramį perduodamas kintamasis „cc“

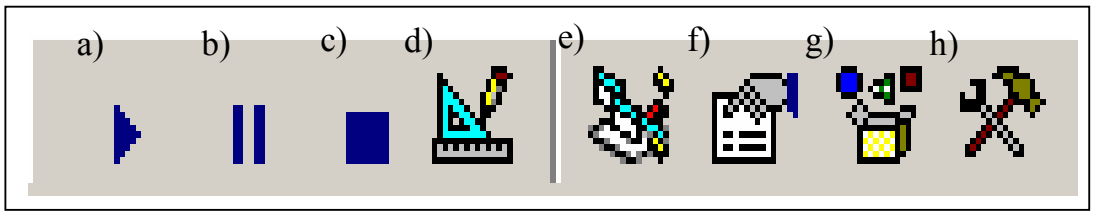
Teksto redaktoriaus lauko aplinkos priedai „Options“. Teksto redaktorius yra aprūpintas projekto kodų sakinių rašymą lengvinančiais priedais. Daug lengviau ir greičiau užrašyti sakinus, kai priedų režimai yra įjungti, todėl jų išjungti nereikia. Ypač praverčia šie priedai pradedantiesiems. Taip išvengiama daug galimų sintaksės klaidų. Pageidaujamą režimą įjungti (išjungti) galima tik išsikvietus dialogo langą „Options“, kurio iškvietimo kelias iš pagrindinio VBA programos meniu **Tools/Options**. Visi režimai surašyti lakšte **Editor/Code Settings**. Yra šeši lengvinantys kodų sakinių rašymą režimai: **Auto Syntax Check** – veikiant šiam režimui, pateikiamas kodų sakinio klaidos aprašas esant galimybei išsamiau paskaityti apie klaidą *Help* bylose; **Require Variable Declaration** – veikiant šiam režimui, teksto redaktoriaus lauko pradžioje atsiras užrašas *Option Explicit*, kuris žymi deklaravimų pradžią; **Auto List Members** – skirtas bruožų sąrašui pateikti teksto redaktoriaus lauke būsimojo bruožo rašymo vietoje; **Auto Quick Info –** skirtas operatoriaus ir funkcijos argumentų su duomenų tipais pateikti teksto redaktoriaus

lauke būsimo operatoriaus arba funkcijos rašymo vietoje; **Auto Data Tips** – kintamojo reikšmei pateikti, kai teksto redaktoriaus lauko žymeklis yra virš kintamojo; **Auto Indent** – tabuliatoriui valdyti.



15 pav. Teksto redaktoriaus aplinkos priedų meniu

Standartinis įrankių meniu (*Toolbar Standard*) turi visas *Microsoft* komandas: išsaugoti projektą (*Save*), objektų ir kodų redagavimo komandas: iškirpti, kopijuoti, padėti (*Cut , Copy, Paste*), surasti teksto redaktoriaus lauke (*Find*), atsisakyti paskutinio veiksmo arba žingsnis atgal (*Undo*) ir žingsnis į priekį (*Redo*). Yra specialių *VBA* redaktoriaus komandų 16. pav.



16 pav. Redaktoriaus komandos: a – programos paleidimas (*Run*), b – pauzė (*Break*), c – nutraukimas (*Reset*), d – programavimo-modeliavimo režimas (*Design Mode*), e – projektų vadovas (*Project Explorer*), f – objektų savybių langas (*Properties Window*), g – objektų bibliotekų sąrašas (*Object Browser*), h – objektų rinkinys (*Toolbox*)

Programos paleidimas (*Run)*, pauzė programos veikimo metu (*Break*), programos nutraukimas (*Reset*), iškviesti programavimo-modeliavimo režimą (*Design Mode*), iškviesti projektų vadovą *(Project Explorer*), iškviesti objektų savybių langą (*Properties Window*), iškviesti objektų bibliotekų sąrašą (*Object Browser*), iškviesti objektų rinkinį (*Toolbox*).

3. Kintamieji ir konstantos

Šiame skyriuje susipažinsime su programavimo kalbos *Visual Basic* kintamųjų ir konstantų įvardijimu, apribojimais, duomenų tipais, matomumu ir deklaravimu.

Kintamasis (*variable*) – tai vardas, kuriuo programuotojas rezervuoja kompiuterio atminties fragmentą, skirtą laikinai programos duomenims saugoti.

Konstanta (*constant*) – tai pastovios reikšmės kintamasis, kurį galima tik skaityti.

Kintamojo ir konstantos vardus programuotojas pasirenka atsižvelgdamas į šias taisykles: vardas prasideda tiktai raide ir neturi būti ilgesnis nei 255 simboliai; vardas nenaudoja skyrybos ir kitų specialių ženklų, išskyrus pabraukimo simbolį; negali sutapti su VB programavimo žodžiais.

Labai svarbu pateikti kintamųjų ir konstantų duomenų tipą. Tai didina programos spartą ir mažina klaidos atsiradimo galimybę. Duomenų tipai su dydžių baitais ir taikymo sritimi pateikti 2 lentelėje.

2 lentelė. Duomenų tipai

Duomenų tipas	Dydis	Sritis
Byte (sveikasis skaičius)	1 baitas	0 iki 255
Boolean (loginis)	2 baitai	*True* (teisinga) arba *False* (neteisinga)
Integer (sveikasis skaičius)	2 baitai	-32,768 iki 32,767
Long (ilgas sveikasis skaičius)	4 baitai	-2,147,483,648 iki 2,147,483,647
Single (viengubo tikslumo su plaukiojančiu kableliu realusis skaičius)	4 baitai	-3.402823E38 iki -1.401298E-45 neigiamosioms reikšmėms; 1.401298E-45 iki 3.402823E38 teigiamosioms reikšmėms
Double (dvigubo tikslumo su plaukiojančiu kableliu realusis skaičius)	8 baitai	-1.79769313486231E308 iki -4.94065645841247E-324 neigiamosioms reikšmėms; 4.94065645841247E-324 iki 1.79769313486232E308 teigiamsioms reikšmėms
Currency (piniginis vienetas)	8 baitai	-922,337,203,685,477.5808 iki 922,337,203,685,477.5807
Decimal (dešimtainis skaičius, kaip *Variant* tipas su funkcija *CDec*)	14 baitų	+/-79,228,162,514,264,337,593,543,950,335 be dešimtainio kablelio; +/-7.9228162514264337593543950335 su 28 vietomis po kablelio į dešinę; mažiausias ne nulinis skaičius yra +/-0.0000000000000000000000000001
Date (data)	8 baitai	*January* 1, 100 *to December* 31, 9999
Object (objektas)	4 baitai	Objekto adresas
String (kintamasis simbolių skaičius)	10 baitų + simbolių skaičius	0 iki apytiksliai 2 Gb
String (fiksuotasis simbolių skaičius)	Simbolių skaičius	1 iki apytiksliai 65,400 baitų
Variant (universalusis skaičiams)	16 baitų	kaip ir *Double*
Variant (universalusis simboliams)	22 baitai + simbolių skaičius	kaip ir *String*
Type (vartotojo duomenų tipas)	Priklauso nuo tipų	Sujungti keli duomenų tipai

Programoje aktualu iš kintamojo vardo žinoti ir duomenų tipą. Todėl kintamojo vardus galime rašyti su priešdėliais (3 lentelė) ir žymėtomis galūnėmis (4 lentelė).

3 lentelė. Kintamojo vardo trijų raidžių priešdėliai

Duomenų tipas	Priešdėlis	Pavyzdys
Boolean	bln	blnTaip
Byte	byt	bytNumeris
Currency	cur	curSuma
Date	dtm	dtmSiandiena
Double	dbl	dblAtsparumas
Integer	int	intEil_numeris
Long	lng	lngSkaičius
Object	obj	objApskritimas
Single	sng	sngPlotis
String	str	strTekstas
User	udt	udtPreke
Variant	vnt	vntKordinate

4 lentelė. Kintamojo vardo žymėtos galūnės

Duomenų tipas	Galūnė	Pavyzdys
Currency	@	suma@
Double	#	atsparumas#
Integer	%	numeris%
Long	&	skaičius&
Single	!	plotis!
String	$	tekstas$

Kintamieji ir konstantos pagal matomumą skirstomi į lokaliuosius, konteinerinius ir globaliuosius. Lokalieji kintamieji arba konstantos matomos tik toje procedūroje, kurioje paskelbtos. Skelbiama pačioje procedūroje ***Dim*** operatoriumi. Konteineriniai matomi tik tame modulyje ir formoje, kurioje paskelbti. Konteineriu vadiname modulio arba formos valdomą programos sritį. Skelbiama prieš modulį arba formą ***Dim*** arba ***Private*** operatoriumi. Globalieji kintamieji arba konstantos matomos visoje programoje. Skelbiama prieš modulį ***Public*** operatoriumi.

Kintamųjų matomumo taikymo galimybės procedūrose pateiktos 5 lentelėje. Kiekvienoje procedūroje yra po vieną atitinkantį procedūros numerį lokalųjį kintamąjį L. Trijų tipų konteineriuose yra trys konteineriniai kintamieji K, kurie perduoda kintamojo informaciją procedūrose esančiame konkrečiame konteineryje. Modulio pradžioje deklaruotas globalusis kintamasis yra vienas ir matomas visose procedūrose.

Prieš naudojant kintamąjį jį reikia deklaruoti: pateikti jo vardą, nurodyti matomumo ir duomenų tipus. Apibendrinta kintamojo deklaravimo išraiška:

[Public | Private | ***Dim****] vardas [****As*** *tipas]*

Čia vardas yra kintamojo vardas, atitinkantis visus reikalavimus, ir tipas pagal 2 lentelę.

Konstanta skelbiama pagal šią sintaksę:

[Public | Private] ***Const*** *vardas [****As*** *tipas] = reikšmė*

Čia vardas ir tipas, kaip deklaruojant kintamąjį, o reikšmė – tai konkreti konstantos informacija.

Kintamojo reikšmė gali būti ne tiktai vienas skaičius ar žodis (sakinys), bet nemažai reikšmių. Tokiems daugiareikšmiams kintamiesiems aprašyti naudojami masyvai. Masyvu (*array*) vadinamas vienodo tipo kintamųjų rinkinys. Jeigu masyvą sudaro skirtingi duomenų tipai, reikia deklaruoti ***Variant*** tipo masyvą. Masyvai yra vienmačiai ir daugiamačiai (iki 60 matavimų). Vienmačių masyvų duomenys surašyti stulpeliu, dvimačių – jie yra lentelėje, o trimačių – trijų matavimų tūryje.

5 lentelė. Kintamųjų matomumas

Forma	Klasės modulis	Modulis
Dim K1 ‘konteinerinis	*Dim K2* ‘konteinerinis	*Public G1* ‘globalusis *Dim K3* ‘konteinerinis
Sub Procedūra_1 *Dim L1* ‘lokalusis *L1 = lokalusis_kintamasis* *K1 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*	*Sub Procedūra_3* *Dim L3* ‘lokalusis *L3 = lokalusis_kintamasis* *K2 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*	*Sub Procedūra_5* *Dim L5* ‘lokalusis *L5 = lokalusis_kintamasis* *K3 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*
Sub Procedūra_2 *Dim L2* ‘lokalusis *L2 = lokalusis_kintamasis* *K1 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*	*Sub Procedūra_4* *Dim L4* ‘lokalusis *L4 = lokalusis_kintamasis* *K2 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*	*Sub Procedūra_6* *Dim L6* ‘lokalusis *L6 = lokalusis_kintamasis* *K3 = konteinerinis_kint* *G1 = globalusis_kintamasis* *……..* *End Sub*

Skiriasi masyvo deklaravimas, nes reikia nurodyti masyvo elementų skaičių. Vienmačio masyvo deklaravimo pavyzdžiai:

Dim** vektorius1 (20) **As Integer

Šiuo atveju pirmojo masyvo elemento indeksas yra „0“, o paskutinio „20“, tai iš viso masyve yra 21 elementas.

Dim** vektorius2 (1 **To** 20) **As Integer

Antruoju atveju deklaruojame pirmojo ir paskutinio masyvo elementų numerius.

Dvimačio masyvo deklaravimo pavyzdys:

Dim** matrica (1 **To** 20, 1 **To** 3) **As Varijant

Čia masyvas yra dvidešimties eilučių, trijų stulpelių ir turinčios įvairios informacijos.

Masyvą, kurio dydis iš anksto nežinomas, galima deklaruoti kaip dinaminį masyvą. Tai atliekama dviem etapais. Pirmiausia konteineryje kintamąjį – masyvą deklaruojame tuščiais skliausteliais, antra, tik procedūroje tam pačiam kintamajam operatoriumi ***ReDim*** nustatome masyvo ribas. Sintaksė:

ReDim [Preserve] *vardas* ***(****matmenys****)*** *[****As*** *tipas] [, vardas* ***(****matmenys****)*** *[****As*** *tipas]]…*

Sintaksės dalys:
preserve – nebūtinas. Raktažodis naudojamas, kai norima išsaugoti anksčiau kintamojo sekai skirtus duomenis keičiant paskutinio sekos deklaravimo sekos talpą;
vardas – būtinas. Kintamojo vardas, atitinkantis visus reikalavimus;
matmenys – būtinas. Kintamojo sekos talpos keitimas. Sintaksė: *[nuo To] iki [,[nuo To] iki]…* Žemutinė sekos riba valdoma operatoriumi *Option Base*. Praleidus sekos žemutinės ribos reikšmę (nurodant tik viršutinę), riba prilyginama nuliui tuo atveju, kai *Option Base* operatorius nenaudojamas;
tipas – nebūtinas. Kintamojo tipas pagal 2 lentelę.

Operatorių ***ReDim*** galima naudoti kelis kartus keičiant masyvų kiekį ar elementų kiekį masyve. Negalime deklaruoti masyvo kintamųjų vienokiu duomenų tipu, o vėliau naudojant ***ReDim*** pakeisti duomenų tipą kitu tuo atveju, kai deklaruojami masyvo kintamieji yra matomumo ribose. Taip būtų naudojant operatorių ***ReDim*** tame pačiame paprogramyje ar funkcijoje arba kai deklaruotas masyvo kintamasis su ***Dim*** operatoriumi.

Naudojant raktažodį ***Preserve***, galima pakeisti tik paskutiniosios sekos kintamajame matmenis išsaugant visų sekų duomenis, negalima keisti matmenis visiems. Pavyzdžiui, deklaruotas dviejų sekų kintamasis vardu „mmm“, kurio pirmosios sekos matmuo yra šeši elementai, o antrosios sekos matmuo yra trys elementai. Visų elementų duomenų tipas yra sveikasis skaičius (*integer*).

ReDim Preserve *mmm(5, 2)* ***As Integer***

Kodėl 5, o ne 6?, ir kodėl 2, o ne 3? Taip yra todėl, kad, praleidus žemutinės masyvo ribos reikšmę, ji imama pagal nutylėjimą 0 arba 1, kai naudotas operatorius *Option Base*.

Kai raktažodis ***Preserve*** nenaudojamas, galima keisti bet kurio masyvo matą, tik tada visi masyvų duomenys bus sunaikinti. Keičiama tik masyvo viršutinė mato reikšmė. Jeigu bandysite pakeisti žemutinę masyvo mato reikšmę, bus klaida.

4. Operatoriai

Visual Basic programavimo kalboje yra priskyrimo, matematiniai, palyginimo ir loginiai operatoriai.

Priskyrimo operatorius (=) skirtas kintamojo išraiškos rezultatui priskirti. Priskyrimo operatorius išsaugo bet kokią reikšmę, kuri gaunama išraiškos dešinėje nuo ženklo *lygu*.

```
kintamasis1 = rezultatas
kintamasis2 = dydis +100
```

Vykdant priskyrimo operaciją, pirmiausia randama reikšmė išraiškos priskyrimo operatoriaus (=) dešinėje, o tik tada rezultatas išsaugomas kintamajame, kurio vardas yra priskyrimo operatoriaus kairėje. Todėl programuojant yra galimos tokios išraiškos:

```
Numeris = Numeris + 1
Suma  = Suma + Tarpinė_suma
```

Matematiniai operatoriai (6 lentelė) naudojami įvairioms išraiškoms rašyti. VB kalboje apostrofo (‘) ženklas reiškia, kad po jo esantis tekstas yra tiktai komentaras.

6 lentelė. Matematiniai operatoriai

Operatorius	Reikšmė	Pavyzdys
+	Sudėtis	a = 5 + 7 ‘ rezultatas a = 12
-	Atimtis	b = 5 – 7 ‘ rezultatas b = –2
*	Daugyba	c = 5 * 7 ‘ rezultatas c = 35
/	Dalyba	d = 15 / 7 ‘ rezultatas d = 2.1428571
\	Sveikoji dalis padalijus	e = 15 \ 7 ‘ rezultatas e = 2
Mod	Liekana padalijus	f = 15 Mod 7 ‘ rezultatas e = 0.1428571
^	Kėlimas laipsniu	g = 5 ^ 3 ‘ rezultatas g = 125
&	Konkatenacija	*tekstas = “viso: “ & suma & “ Lt“* ‘ rezultatas tekstas = “viso: 47.75 Lt“ jeigu kintamasis suma = 47.75

VB programose galime rašyti matematines išraiškas, kuriose yra ne tiktai skaičiai ir matematiniai operatoriai, bet ir kintamieji, funkcijos (7 skyrius) ir skliausteliai.

```
a = 12
b = 7 * (5 + 2 * a ) + 3          ‘ rezultatas b = 206
c = 7 * (5 + 2 * (a  + b) + 3     ‘ rezultatas c = 3108
d = (b ^ 0.5 + c ^ 4 ) ^ 0.25     ‘ rezultatas d = 12.00207595
e = (5 + b ) / (7 + a)            ‘ rezultatas e = 11.10526
f = b – d * e / c                 ‘ rezultatas f = 205.9571151
g = (5 + a ) * (7 + b) / 2        ‘ rezultatas g = 1810.5
```

Palyginimo operatoriai (7 lentelė) naudojami valdymo struktūrose (6-ajame skyriuje). Šie operatoriai kintamuosius ir reiškinius leidžia palyginti.

7 lentelė. Palyginimo operatoriai

Operatorius	Reikšmė	Pavyzdys
=	Lygu	*d = 10*
<>	Nelygu	*vardas <> ”Simas“*
<	Mažiau nei	*c < 50*
>	Daugiau nei	*d > h*
<=	Mažiau arba lygu	*a <= 156*
>=	Daugiau arba lygu	*laikas >= ”10:30“*
Is	Objektų lyginimas	*objAA Is objBB*
Like	Teksto (stringo) lyginimas	*”aBBBa“* ***Like*** *”a*a“* ‘ pateikia *True* *”F“* ***Like*** *”[A-Z]“* ‘ pateikia *True* *”a2a“* ***Like*** *”a#a“* ‘ pateikia *True* *”BAT123“* ***Like*** *”B?T*“* ‘ pateikia *True*

Loginiai operatoriai (8 lentelė) naudojami valdymo struktūrose (6-ajame skyriuje). Šie operatoriai leidžia apibendrinti kelių palyginimo reiškinių rezultatus ir sukurti sudėtingus kriterijus, naudojamus procedūrose.

8 lentelė. Loginiai operatoriai

Operatorius	Reikšmė	Pavyzdys
And	Konjunkcija	***p1 And p2*** ‘ *True*, jeigu p1 ir p2 lygūs *True*, priešingu atveju – *False*
Or	Disjunkcija	***p1 Or p2*** ‘ *True*, jeigu nors vienas arba abu p1, p2 lygūs *True*, priešingu atveju – *False*
Not	Inversija	***Not p1*** ‘ *True*, jeigu p1 lygus *False*, priešingu atveju – True
Xor	Pasirinkimas	***p1 Xor p2*** ‘ *True*, jeigu p1 nelygus p2, priešingu atveju – *False*
Eqv	Ekvivalentiškumas	***p1 Eqv p2*** ‘ *True*, jeigu p1 lygus p2, priešingu atveju **–** *False*
Imp	Inplikacija	***p1 Imp p2*** ‘ *False*, jeigu p1 lygus *True*, o p2 lygus *False*, visi kiti atvejai – *True*

Sudėtinėse išraiškose operacijos daromos pagal nustatytą prioritetinę seką (9 lentelė). Pirmiausia operacijos daromos skliausteliuose. Jeigu operatoriai yra to paties prioriteto, tai atliekami iš eilės iš kairės į dešinę. Daugybos ir dalybos matematiniai operatoriai yra to paties prioriteto. Palyginimo operatoriai irgi tarpusavyje yra to paties prioriteto. Atlikus matematinius veiksmus, dalys sujungiamos (konkatenacija), o toliau vykdomos loginės operacijos.

9 lentelė. Operatorių prioritetai

Operatoriaus	Prioritetas
(..)	Reiškinys skliausteliuose
^	Kėlimas laipsniu
-	Neigiamosios reikšmės priskyrimas
* /	Daugyba ir dalyba
\	Sveikoji dalis padalijus
Mod	Dalybos liekana
+ -	Sudėtis, atimtis
&	Sujungimas (konkatenacija)
<,<=,>,>=,<>,=, *Is, Like*	Palyginimo operatoriai yra to paties prioriteto ir operacijos vykdomos iš kairės į dešinę
Not	Inversija
And	Konjunkcija
Or	Disjunkcija
Xor	Pasirinkimas
Eqv	Ekvivalentiškumas
Imp	Emplikacija

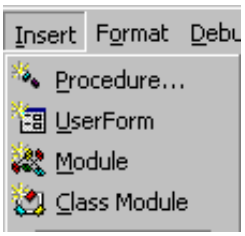
5. Procedūros

Visual Basic programa susideda iš procedūrų, kurios atlieka konkrečius, nepriklausomus nuo kitos programos dalies veiksmus. Procedūra parašoma ir išbandoma atskirai nuo kitų programos elementų. Procedūros labai naudingos, kai dažnai atliekami tie patys veiksmai, nes procedūrą galime pakartoti kiek reikia kartų. Procedūros sintaksė:

```
Public | Private  Sub procedūros_vardas_[įvykis] ([argumentai])
        procedūros tekstas
End Sub
```

Procedūros rašomos moduliuose ir formose. Procedūrų matomumas yra dviejų tipų: globalusis (*Public*), matomas visoje programoje, arba lokalusis (*Private*), matomas konkrečiame konteineryje. Procedūros vardas gali būti susietas su įvykiu, kuriam atsitikus procedūra pradeda veikti, pavyzdžiui, paspaudus konkrečios procedūros aktyvinimo mygtuką. Procedūros gali būti be argumentų arba su vienu ar keliais argumentais.

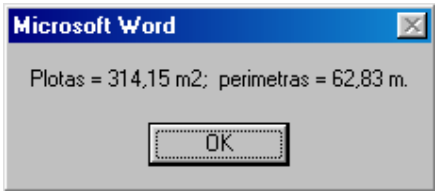
VB redaktoriaus meniu ***Insert*** komandose matome procedūros kūrimo komandą (17 pav.) ir formos bei modulių pradėjimo komandas. Pirmiausia susikuriame formą arba modulį, o jau jame rašome procedūras.



17 pav. Meniu *Insert*

Pateikiamas pavyzdys lokaliosios procedūros be argumentų esant lokaliajam kintamųjų deklaravimui ir duomenų išvedimo funkcijai *MsgBox*. Šio pavyzdžio rezultatai pateikti 18 pav.

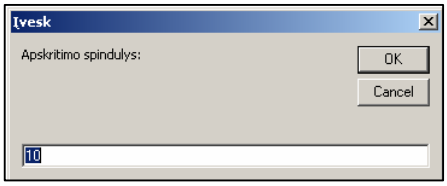
```
Private Sub Procedūra1()
   Dim spindulys, plotas, perimetras
   Const pi = 3.1415
   spindulys = 10#
   plotas = pi * spindulys ^ 2
   perimetras = 2 * pi * spindulys
   MsgBox "Plotas = " & plotas & " m2; " & _
   " perimetras = " & perimetras & " m."
End Sub
```



18 pav. Procedūros1 rezultatas

Pateikiamas lokaliosios procedūros su argumentais ir konteineriniu kintamųjų deklaravimu modulyje pavyzdys. Šiame modulyje parašytos dvi procedūros, kurių kintamieji ir konstanta deklaruoti modulio pradžioje, t. y. konteinerinis deklaravimas. Programos tekste po apostrofo (‘) simbolio rašomi komentarai. Pirmoji procedūra „Apskritimas“ globali ir turi tris argumentus. Antroji procedūra įvedus apskritimo spindulį (19 pav.) iškviečia procedūrą „Apskritimas“, kuri suranda kitus du argumentus (plotas ir perimetras). Toliau „Procedūra_2“ iškviečia pranešimo langą, kuriame pateikia gautus rezultatus.

```
Option Explicit
   ‘ Kintamųjų deklaravimas modulyje
   Dim spindulys, plotas, perimetras
   Const pi = 3.1415

Public Sub Apskritimas(spindulys, plotas, perimetras)
   ‘ Skaičiuoja apskritimo plotą ir perimetrą.
   plotas = pi * spindulys ^ 2
   perimetras = 2 * pi * spindulys
End Sub
```

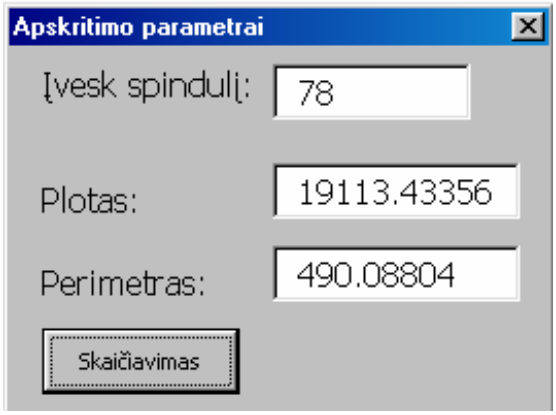


19 pav. Įvedimo langas

```
Private Sub Procedūra_2()
   ‘ Įvedus spindulį, gaunamas apskritimo plotas ir perimetras.
   spindulys = InputBox("Apskritimo spindulys:", "Įvesk", 10)
   Apskritimas spindulys, plotas, perimetras
   MsgBox "Plotas = " & plotas & " m2; " & " perimetras = " & perimetras & " m."
End Sub
```

Pateikiamas globaliosios procedūros pavyzdys su argumentais ir globaliuoju kintamųjų deklaravimu modulyje bei vartotojo formos naudojimu. Šiame pavyzdyje parašytos dvi procedūros, kurių kintamieji ir konstanta deklaruoti modulio pradžioje. Pirmoji procedūra „Apskritimas“ yra modulyje. Antroji procedūra yra formoje ir suieta su įvykiu – mygtuko „Skaičiavimas“ paspaudimu. Pačioje formoje parašomas apskritimo spindulys (20 pav.) ir paspaudus mygtuką iškviečiama procedūra „cmbSkaičiavimas“. Taip surandamas plotas ir perimetras. Formoje pamatome gautus rezultatus.

```
Option Explicit

   Public spindulys, plotas, perimetras
   Const pi = 3.1415

Public Sub Apskritimas(spindulys, plotas, perimetras)
   plotas = pi * spindulys ^ 2
   perimetras = 2 * pi * spindulys
End Sub

‘ Formos mygtukas Skaičiavimas
Private Sub cmbSkaičiavimas_Click()
spindulys = TextBox1.Value
Apskritimas spindulys, plotas, perimetras
TextBox2.Value = plotas
TextBox3.Value = perimetras
End Sub
```



20 pav. Forma su mygtuku

6. Valdymo struktūros

Valdymo struktūros leidžia parašyti sudėtingas programas atliekančias įvairias funkcijas ir net darančias savarankiškus sprendimus. Atsižvelgiant į matematinių reiškinių rezultatus, kintamųjų reikšmes ar programos vartotojo komandas, galima keisti programos veiksmus. *Visual Basic* programavimo kalbos valdymo struktūras sudaro: sąlyginės, besąlyginės ir ciklinės valdymo struktūros.

Sąlyginės valdymo struktūros įvertina pateiktą loginę sąlygą ir, atsižvelgiant į rezultatą, jeigu teigiamasis, (*True*) vykdo kitą operatorių arba jeigu neigiamasis, (*False*) pereina prie programos operatoriaus esančio už šios struktūros. Paprasčiausia sąlygos struktūra yra operatorius ***If …Then***, kurio sintaksė ir pavyzdys:

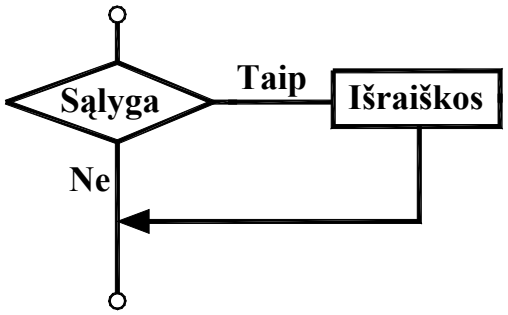
If *sąlyga* ***Then*** *išraiška*	*If temperatūra >= 50 Then tekstas = ”Per karšta !“*

čia sąlyga – bet koks loginis reiškinys, o išraiška – bet kokia matematinė ar programinė išraiška, parašyta toje pačioje eilutėje.

Kitas šios sąlygos išplėstinis variantas. Sintaksė ir pavyzdys:

If *sąlyga* ***Then***	*If temperatūra > 50 Then*
išraiškos	*tekstas = ”Per karšta !“*
End If	*kintamasis = temperatūra – 50*
	End If

čia išraiškos gali užimti kelias eilutes.

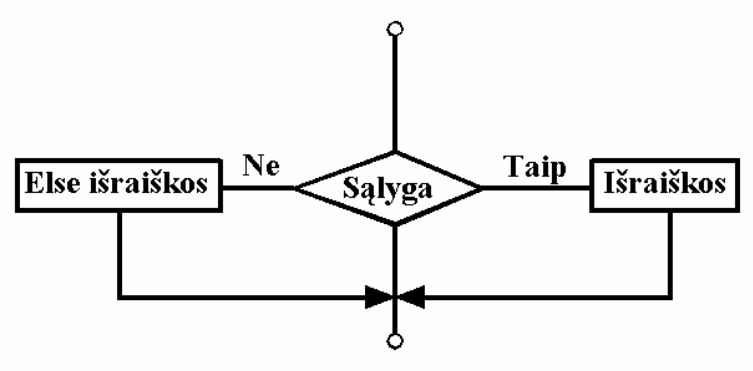


21 pav. Operatoriaus *If...Then* veikimo blokinė schema

Sudėtingesnė sąlygos struktūra yra operatorius ***If…Then…Else***, kurio sintaksė ir pavyzdys:

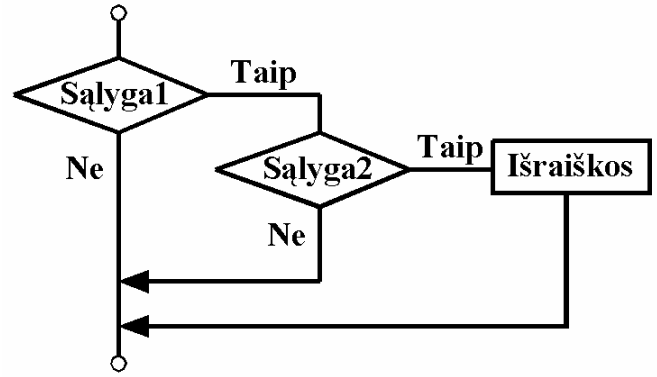
If *sąlyga* ***Then***	*If temperatūra > 30 Then*
išraiškos	*tekstas1 = ”Per karšta !“*
Else	
else išraiškos	*tekstas2 = ”Normalu !“*
End If	*End If*

čia sąlyga – bet koks loginis reiškinys, o išraiškos – bet kokios programinės išraiškos, atitinkančios teigiamąjį loginį reiškinį ir parašytos vienoje ar keliose eilutėse, *Else* išraiškos – bet kokios programinės išraiškos, atitinkančios neigiamąjį loginį reiškinį ir parašytos vienoje ar keliose eilutėse.



22 pav. Operatoriaus *If…Then…Else* veikimo blokinė schema

Prireikus operatorius ***If…Then*** arba ***If…Then…Else*** galima naudoti vienas kito viduje sukuriant sudedamąją sąlygos struktūrą.



23 pav. Sudėtinė operatorių *If…Then* veikimo blokinė schema

Dar sudėtingesnė sąlygos struktūra yra operatorius ***If…Then…ElseIf***, kurio sintaksė ir pavyzdys pateikti atitinkamais stulpeliais:

If *sąlyga1* ***Then***	*If temperatura > 27 Then*
išraiškos	*tekstas1 = ”Per karsta !“*
ElseIf *sąlyga2*	*ElseIf temperatura < 10*
Išraiška2	*tekstas2 = ”Per šalta !“*
……	
[***Else***	*Else*
else išraiškos]	*tekstas3 = ”Noramalu !“*
End If	*End If*

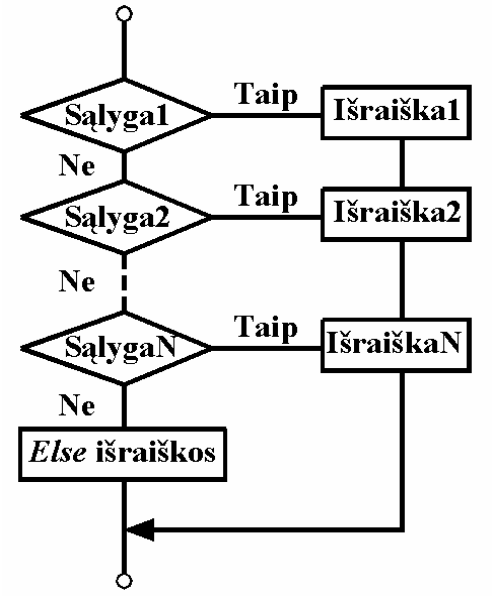
Čia *sąlyga1, sąlyga2* – bet kokie loginiai reiškiniai, o išraiškos – bet kokios programinės išraiškos, atitinkančios teigiamąjį loginį reiškinį, suformuotą pirmąja *sąlyga1*. Jeigu pirmoji sąlyga teikia neigiamą atsakymą (*False*), tada pereinama prie operatoriaus ***ElseIf*** ir antrosios *sąlygos2*. Jeigu *sąlyga2* pateikia teigiamą atsakymą (*True*), tai vykdomos ***ElseIf*** išraiškos. Jeigu abi sąlygos pateikia neigiamą rezultatą (*False*), galima naudoti operatorių ***Else*** (laužtiniuose skliaustuose). Vidinį operatorių ***ElseIf*** galima kartoti kelis kartus, tai matome 24 pav.

Kada pasirinkimo variantų daug, o kintamasis vienas, tai geriausia sąlygos struktūra yra operatorius ***Select Case***, kurio sintaksė ir pavyzdys:

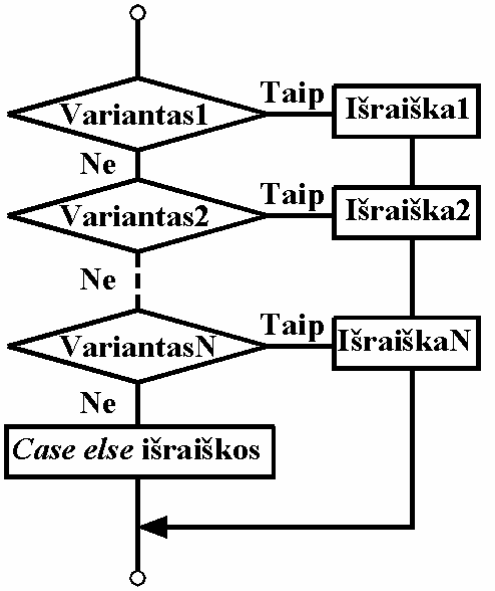
```
Select Case išraiška
  Case variantas1
        išraiška1
  Case variantas2
        išraiška2
……….
  Case variantasN
        išraiškaN
 [ Case Else
        else išraiška ]
End Select
```

```
Select Case tekstas
  Case > 40
        tekstas = ”Labai karšta !“
  Case > 30
        tekstas = ”Karšta !“
  Case > 20
        tekstas = ”Noramalu !“
  Case > 10
        tekstas = ”Nešalta !“
  Case Else
        tekstas = ”Šalta !“
End Select
```

čia išraiška – reiškinys su kintamuoju arba tiktai kintamasis, o variantai, bet kokios programinės išraiškos, atitinkančios teigiamąjį loginį reiškinį ir parašytos viena ar keliomis eilutėmis, ***Else*** išraiškos – bet kokios programinės išraiškos, atitinkančios neigiamąjį loginį reiškinį ir parašytos viena ar keliomis eilutėmis (25 pav.).



24 pav. Operatoriaus *If…Then…ElseIf* veikimo schema



25 pav. Operatoriaus *Select Case* veikimo schema

Besąlyginės valdymo struktūros keičia nuoseklią operatorių vykdymo seką neįvertinant jokių sąlygų ir paprasčiausiai tęsia programos vykdymą iš nurodytos vietos. Besąlyginė valdymo struktūra yra operatorius ***GoTo***, kurio sintaksė ir pavyzdys pateikti stulpeliais:

```
GoTo žymė
……..
žymė:
```

```
GoTo  aa
……
aa:
```

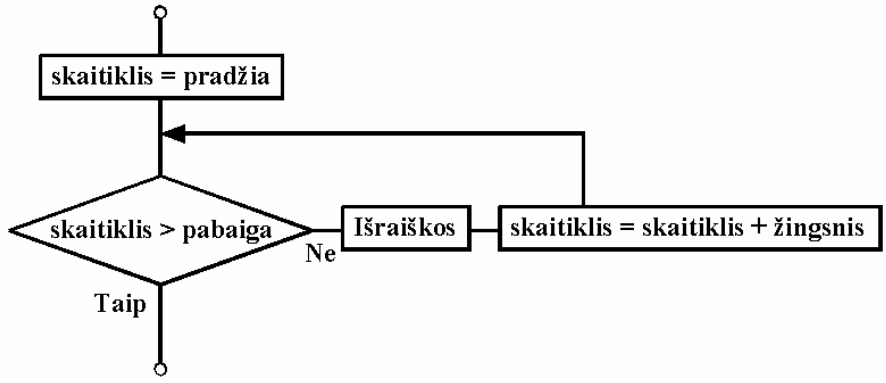
čia žymė – bet koks žodis, kuris nurodo kitą programos eilutę. Žymė gali būti aukščiau arba žemiau ***GoTo*** operatoriaus.

Ciklinės valdymo struktūros kartoja veiksmus nurodytą kartų skaičių arba tol, kol tiks tam tikra sąlyga. Pasikartojančių veiksmų atlikimas cikle vadinamas ciklo iteracija. Ciklai yra keturių tipų.

Pirmas ciklo tipas ***For…Next*** naudojamas, kai yra žinomas ciklo iteracijų skaičius. Šio ciklo sintaksė ir pavyzdys:

For *skaitiklis = pradžia* ***To*** *pabaiga [****Step*** *žingsnis]*	*For i = 1 To 10*
išraiškos	*s = s + i*
Next *[skaitiklis]*	*Next i*

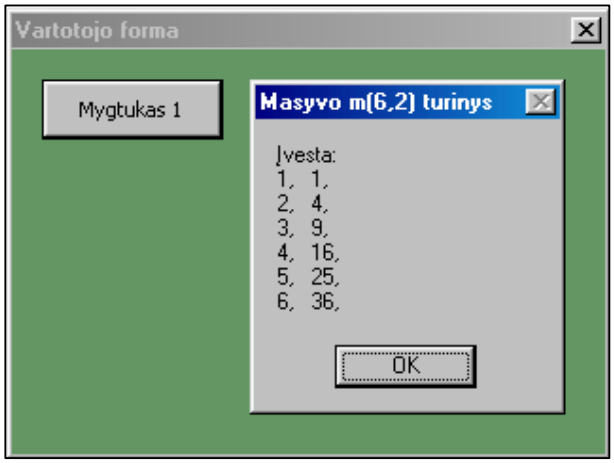
Čia skaitiklis – bet kokio skaitmeninio tipo ciklo pasikartojimų kintamasis. Tinka sveikasis skaičius (*Integer*) jeigu žingsnis yra sveikasis skaičius, o iteracijų mažiau 32767. Nurodomos skaitiklio pradinė (pradžia) ir galinė (pabaiga) reikšmės. Jeigu iteracijos žingsnis lygus vienetui, tai žingsnį nurodyti nebūtina. Priešingu atveju reikalingas operatorius ***Step*** ir žingsnio reikšmė.



26 pav. Ciklo *For...Next* veikimo blokinė schema

Pavyzdys:

```
Private Sub cmbMygtukas1_Click()
'Suformuojame masyvą M(6,2)
For i = 1 To 6
   For j = 1 To 2
      m(i, j) = i ^ j
   Next j
  Next i
t = ""
For i = 1 To 6
   For j = 1 To 2
      t = t & m(i, j) & ",   "
   Next j
      t = t & Chr(13)
Next i
MsgBox "Įvesta: " & Chr(13) & t, , _
     "Masyvo m(6,2) turinys"
End Sub
```



27 pav. Vartotojo forma su pranešimo langu

Pavyzdyje funkcijos ***MsgBox*** eilutė yra užrašyta dviejomis eilutėmis, kurias jungia programos eilutės perkėlimo ženklas (_) pabraukimas. Simboliu dvitaškis (:), atvirkščiai, galima kelias programos eilutes užrašyti viena.

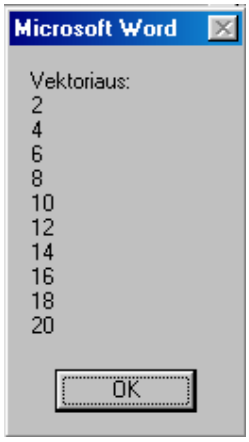
Antras ciklo tipas ***For Each…Next*** naudojamas, kai yra žinomas konkrečios grupės elementų sąrašas tokioje kaip objektų kolekcija arba elementai vektoriuje ir matricoje. Čia nėra ciklų skaitiklio. Šio ciklo sintaksė ir pavyzdys:

For Each *elementas* ***In*** *grupė*	*For Each x In aa*
išraiškos	*x = x + 5*
Next *[elementas]*	*Next*

Čia elementas yra grupės narys. Jeigu grupė yra objektų kolekcija, tai elementas turi būti kintamasis ***Variant*** arba ***Object***. Jeigu grupė yra vektorius ar matrica, tai elementas turi būti kintamasis ***Variant.***

Pavyzdžiai:

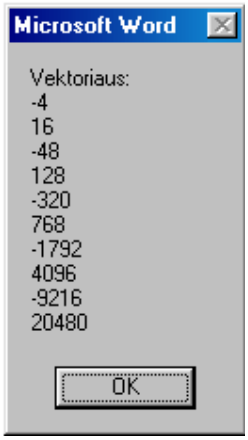
```
Option Explicit
Dim vek(10) As Variant

Public Sub Vektorius(vek)
Dim i As Integer
Dim tek As String
tek = ""
For i = 1 To 10
    tek = tek & vek(i) & Chr(13)
Next i
MsgBox "Vektoriaus: " & Chr(13) & tek
End Sub
```



28 pav. Pranešimo langas su rezultatais

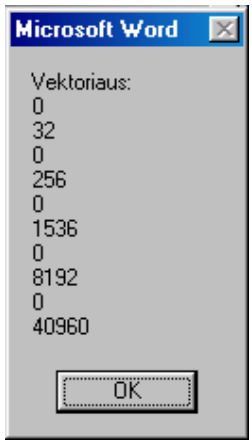
Forma:

```
Option Explicit
Dim i As Integer
Dim v(10) As Variant

Public Sub cmbVV_Click()
 Dim x As Variant
 Dim vv(10) As Double

For i = 1 To 10
   vv(i) = i + i
Next i
Vektorius vv
 i = 0
 For Each x In vv
   x = (–2) ^ i * x
   v(i) = x
   i = i + 1
 Next x
Vektorius v
End Sub
```



29 pav. Vektorius VV

```
Public Sub cmbVVV_Click()
Dim y As Variant
Dim vvv(10) As Double
i = 0
 For Each y In v
   If y <= 0 Then
         y = 0
     Else
         y = y * 2
     End If
          vvv(i) = y
         i = i + 1
   Next y
Vektorius vvv
End Sub
```



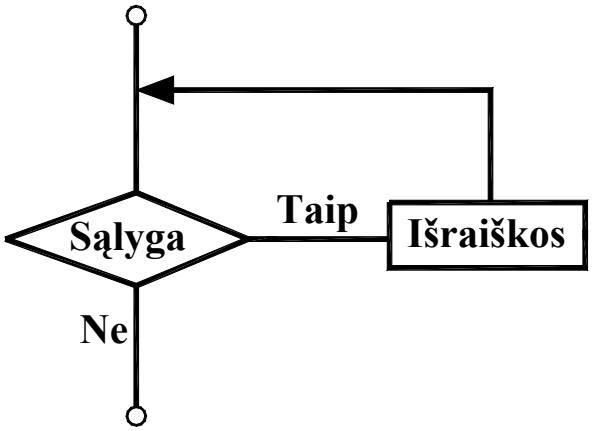
30 pav. Vektorius VVV

Trečias ciklo tipas ***Do...Loop*** naudojamas, kai reikalinga tikrinimo sąlyga. Yra keturi šio ciklo variantai, kuriuose tikrinama sąlyga ir, atsižvelgiant į gautą rezultatą, ciklas tęsiamas arba baigiamas. Sąlyga gali įgyti teisingą (*True*) arba klaidingą (*False*) reikšmę. Pirmo varianto ciklo sintaksė ir pavyzdys:

```
Do While sąlyga
  išraiškos
Loop
```

```
Do While p > 10
 p = p – 1
 s = s + 1
Loop
```

Tai ciklas, kada sąlyga tikrinama pradžioje ir išraiškos atliekamos, jeigu sąlyga yra teisinga. Jeigu sąlyga klaidinga, ciklas nevyksta ir išeinama iš ciklo.



31 pav. Ciklo *Do While…Loop* veikimo blokinė schema

Pavyzdys:

```
Private Sub DoWhile_Click()
  s = 0
  p = InputBox("Pradinis skaičius", "Įvesk", 20)
  Do While p > 10
     p = p - 1
     s = s + 1
  Loop
```

```
    MsgBox "Atlikta " & s & " pakartojimu."
End Sub
```

Antro varianto ciklo sintaksė ir pavyzdys:

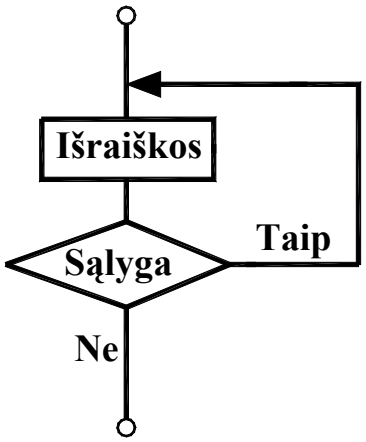
Do Until *sąlyga*	*Do Until p = 10*
išraiškos	*p = p – 1*
Loop	*s = s + 1*
	Loop

Tai ciklas, kada sąlyga tikrinama iš pradžių ir atliekamas, jeigu sąlyga yra klaidinga. Jeigu sąlyga tampa teisinga, ciklas nevykdomas ir išeinama iš jo.

Trečio varianto ciklo sintaksė ir pavyzdys:

Do	*Do*
išraiškos	*p = p – 1*
Loop While *sąlyga*	*s = s + 1*
	Loop While p > 10

Tai ciklas, kada sąlyga tikrinama pabaigoje, ir atliekamas, jeigu sąlyga yra teisinga. Jeigu sąlyga tampa klaidinga, ciklas nevykdomas ir išeinama iš jo.



32 pav. Ciklo *Do…Loop While* veikimo blokinė schema

Ketvirto varianto ciklo sintaksė ir pavyzdys:

Do	*Do*
išraiškos	*p = p – 1*
Loop Until *sąlyga*	*s = s + 1*
	Loop Until p = 10

Tai ciklas, kada sąlyga tikrinama pabaigoje, ir atliekamas, jeigu sąlyga yra klaidinga. Jeigu sąlyga tampa teisinga, ciklas nevykdomas ir išeinama iš jo.

7. Funkcijos

Visual Basic programavimo kalboje naudojamos vartotojo ir vidinės funkcijos. Funkcija skirtingai nei procedūros grąžina rezultatą, todėl jį reikia priskirti kintamajam arba naudoti kaip kitų funkcijų ar procedūrų argumentą. Funkcijos labai naudingos, kai dažnai atliekami tie patys skaičiavimai ir veiksmai. Funkciją galime išsikviesti bet kurioje programos vietoje ir kiek norime kartų. Vartotojo funkcijos sintaksė:

[Public | Private] ***Function*** *funkcijos_vardas* ***(****[argumentai]****)*** *[* ***As*** *tipas****]***
funkcijos tekstas
[funkcijos_vardas = išraiška]
End Function

Funkcijų matomumas yra dviejų tipų: globalusis (*Public*), matomas visoje programoje, ir lokalusis (*Private*), matomas konkrečiame konteineryje. Funkcijos vardas parašomas pagal kintamųjų įvardijimo taisykles. Funkcijos gali būti be argumentų arba su vienu ar keliais argumentais, atskirtais kableliu. Nurodomas funkcijos grąžinamos reikšmės tipas, bet kuris iš 2 lentelės. Priešpaskutinė eilutė, parodyta laužtiniuose skliausteliuose, reiškia, kad eilutė yra nebūtina, tačiau, reikia pabrėžti, kad labai pageidautina.

Susipažinkime su pagrindinėmis vidinėmis funkcijomis.

10 lentelė. Matematinės funkcijos

Sintaksė	Apibūdinimas	Argumentas
Abs *(skaičius)*	Suranda argumento absoliutinį dydį	
Atn *(skaičius)*	Skaičiuoja argumento arctangentą	$-\pi/2 \div \pi/2$
Cos *(kampas)*	Skaičiuoja argumento kampas kosinusą	radianais
Exp *(skaičius)*	Kelia skaičių e argumento laipsniu	<= 70+E10
Fix *(skaičius)*	Artimiausias sveikasis skaičius, bet ne mažesnis už argumentą	
Int *(skaičius)*	Artimiausias sveikasis skaičius, bet ne didesnis už argumentą	
Log *(skaičius)*	Skaičiuoja argumento natūrinį logaritmą	>= 0
Rnd *[(skaičius)]*	Pateikia atsitiktinį skaičių nuo 0 iki 1	
Round *(a [, n])*	a – apvalinama skaitmeninė išraiška, n – skaitmenų skaičius po kablelio. Jeigu n praleistas, tai apvalinama iki sveikojo skaičiaus	
Sgn *(skaičius)*	Nustato argumento ženklą. Jeigu argumentas teigiamasis, tai pateikia 1, jeigu lygus nuliui, tai 0, jeigu neigiamasis, tai –1	
Sin *(kampas)*	Skaičiuoja argumento sinusą	radianais
Sqr *(skaičius)*	Skaičiuoja argumento kvadratinę šaknį	>= 0
Tan *(kampas)*	Skaičiuoja argumento tangentą	radianais

11 lentelė. Masyvų valdymo funkcijos

Sintaksė	Apibūdinimas
Array *(sąrašas)*	Pateikia vienmatį masyvą su *Variant* elementais, atitinkančiais argumento sąrašas narius
Choose *(indeksas, p1[, p2, ...])*	Pateikia pasirinkimo pi *Variant* reikšmę, atsižvelgiant į indekso eilės numerį
IsArray *(kintamasis)*	Patikrina, ar kintamasis yra masyvas. Jei masyvas pateikia *True*, priešingu atveju – *False*
Lbound *(masyvas [,matmenų eilės numeris])*	Suranda masyvo mažiausiąjį galimą elemento indeksą. Jeigu nenurodyti masyvo matmenys, tai elemento indeksas imamas lygus 0 arba 1 pagal *Option Base* režimą. Pvz.: *Dim M(1 To 100, 0 To 3, -3 To 7)* *LBound(M,1)* ‘ pateikia 1 *LBound(M,2)* ‘ pateikia 0 *LBound(M,3)* ‘ pateikia -3

Ubound *(masyvas [,matmenų eilės numeris])*	Suranda masyvo didžiausiąjį galimą elemento indeksą. Jeigu nenurodyti masyvo matmenys, tai elemento indeksas imamas lygus 0 arba 1 pagal *Option Base* režimą. Pvz.: *Dim M(1 To 100, 0 To 3, -3 To 7)* *UBound(M,1)* ‘ pateikia 100 *UBound(M,2)* ‘ pateikia 3 *UBound(M,3)* ‘ pateikia 7

12 lentelė. Teksto valdymo funkcijos

Sintaksė	Apibūdinimas
InStr *([pradžia,] ieškomas, tekstas [, konstanta])*	Suranda simbolio poziciją. Pradžia – paieškos pradžios pozicija. Ieškomas – ieškomas *stringas*. Tekstas – eilutė, kurioje vyksta paieška. Konstanta imama iš 13 lentelės. Pvz.: *Tekstas = ”xxpxxpxxPxxP“* *InStr(1, ”P“, Tekstas, 1)* ‘ pateikia 9 *InStr(4, ”P“, Tekstas, 1)* ‘ pateikia 6 *InStr(1, ”p“, Tekstas, 1)* ‘ pateikia 3
InStrRev *(ieškomas, tekstas [,pradžia [, konstanta]])*	Suranda simbolio poziciją skaičiuojant nuo teksto pabaigos. Ieškomas – ieškomas *stringas*. Tekstas – eilutė, kurioje vykdoma paieška. Pradžia – paieškos pradžios pozicija, jeigu nenurodyta, tai ieškoma nuo paskutinio simbolio. Konstanta imama iš 13 lentelės
Join *(masyvas[, ženklas])*	Vienmačio masyvo elementų sujungimas. Ženklas, kuris naudojamas tarp elementų. Jeigu ženklas praleistas, tai paliekamas tarpelis
Left *(tekstas, ilgis)*	Pateikia teksto dalį, kurios simbolių skaičius lygus nurodytam ilgiui. Pvz.: *Tekstas = ”Sveikas, pasauli“* *Left (Tekstas, 1)* ‘ pateikia ”S“ *Left (Tekstas, 7)* ‘ pateikia ”Sveikas“
Len *(tekstas | kintamasis)*	Suranda teksto simbolių skaičių arba kintamojo dydį baitais
Lcase *(tekstas)* ***Ucase*** *(tekstas)*	Parašo tekstą mažosiomis (*LCase*), didžiosiomis (*UCase*) raidėmis
Ltrim *(tekstas)* ***Rtrim*** *(tekstas)* ***Trim*** *(tekstas)*	Panaikina tekste tarpelius iš kairės (*LTrim*), iš dešinės (*RTrim*), iš abiejų pusių (*Trim*). Pvz.: *Tekstas = ” Sveikas “* *LTrim (Tekstas)* ‘ pateikia ”Sveikas “ *RTrim (Tekstas)* ‘ pateikia ” Sveikas“ *Trim (Tekstas)* ‘ pateikia ”Sveikas“
Mid *(tekstas, pradžia [, ilgis)*	Pateikia nurodytą teksto dalį. Nurodome pradžią ir ilgį – simbolių skaičių. Pvz.: *Tekstas = ”Labas rytas“* *Mid (Tekstas, 1, 5)* ‘ pateikia ”Labas“ *Mid (Tekstas, 7, 5)* ‘ pateikia ”rytas“
Replace *(išraiška, rasti, keisti [,pradžia[, suma[, konstanta]]])*	Pakeičia nurodytą teksto dalį. Išraiška – teksto kintamasis; rasti – keičiamas simbolių rinkinys; keisti – naujas simbolių rinkinys. Nebūtini parametrai: pradžia – paieškos pradžios simbolio numeris; suma – pasikeitimų skaičius (nepateikus prilyginamas 1 ir atliekami visi galimi keitimai); konstanta imama iš 13 lentelės

Right *(tekstas, ilgis)*	Pateikia teksto iš dešinės dalį, kurios simbolių skaičius lygus nurodytam ilgiui
Space *(skaičius)*	Pateikia nurodytą skaičių tarpelių. Pvz.: *Tekstas = ”Sveikas,“ & Space(10) & ”pasauli“*
StrComp *(tekstas1, tekstas2 [, konstanta)*	Palygina du tekstus, kur nebūtina konstanta iš 13 lentelės. Rezultatai: –1, 1 tekstas trumpesnis už 2 tekstą; 0, 1 tekstas lygus 2 tekstui; 1, 1 tekstas ilgesnis už 2 tekstą; *Null*, 1 teksto arba 2 teksto nėra
StrConv *(tekstas, konstanta)*	Pakeičia tekstą, atsižvelgiant į teksto valdymo konstantą (14 lentelė)
String *(skaičius, simbolis)*	Pateikia eilutę su simboliu, pakartotu nurodytu skaičiumi. Pvz.: *String (5, ”*“)* ‘ pateikia ”*****“ *String (3, 42)* ‘ pateikia ”***“ *String (4, ”ABC“)* ‘ pateikia ”AAAA“

13 lentelė. Palyginimo konstantos

Konstanta	Reikšmė	Apibūdinimas
vbUseCompareOption	–1	Lyginimo metodas, nurodytas *Option Compare* operatoriumi
vbBinaryCompare	0	Lyginami dvejetainiai kodai
vbTextCompare	1	Lyginami simboliai
vbDatabaseCompare	2	Lyginama tiktai *MS Access*

14 lentelė. Teksto valdymo konstantos

Konstanta	Reikšmė	Apibūdinimas
vbUpperCase	1	Didžiosios raidės
vbLowerCase	2	Mažosios raidės
vbProperCase	3	Žodžio pirmoji raidė yra didžioji
vbUnicode	64	Tekstą pakeičia unikodu (ISO simbolių standartas) pagal sistemos aktyvią kodų lentelę
vbFromUnicode	128	Tekstą pakeičia iš unikodo (ISO simbolių standartas) į sistemos aktyvius kodus

15 lentelė. Duomenų tipo pakeitimo funkcijos

Sintaksė	Argumentas	Gaunamas duomenų tipas
Cbool *(išraiška)*	Teksto eilutė arba skaitmeninė išraiška	*Boolean*
Cbyte *(išraiška)*	Sveikasis skaičius nuo 0 iki 255	*Byte*
Ccur *(išraiška)*	Bet koks skaičius nuo –922,337,203,685,477.5808 iki 922,337,203,685,477.5807	*Currency*
Cdate *(išraiška)*	Datos išraiška	*Date*
CDbl *(išraiška)*	Bet koks skaičius	*Double*
Cdec *(išraiška)*	Bet koks skaičius	*Decimal*
Cint *(išraiška)*	Bet koks skaičius nuo –32 768 iki 32 767	*Integer*
CLng *(išraiška)*	Bet koks skaičius nuo –2,147,483,648 iki 2,147,483,647	*Long*
CSng *(išraiška)*	Bet koks skaičius duomenų tipo ribose	*Single*

CStr *(išraiška)*	Teksto eilutė arba skaitmeninė išraiška	*String*
Cvar *(išraiška)*	Kaip *Double* skaičiams ir kaip *String* tekstui	*Variant*
CVErr *(išraiška)*	Bet kokiam klaidos kodui	*Variant*
Str *(skaičius)*	Bet koks skaičius. Prieš teigiamąjį skaičių reikalingas tarpelis	*String*
Val *(tekstas)*	Skaičiai užrašyti kaip teksto eilutė. Panaikina tarpelius, *tab* ir *enter* ženklus. Pvz: *Val ("7812")* ' pateikia 7812 *Val (" 7 8 1 2 ")* ' pateikia 7812 *Val ("78 ir 12")* ' pateikia 78	
TypeName *(kintamasis)*	Pateikia kintamojo tipo pavadinimą	
VarType *(kintamasis)*	Pateikia kintamojo duomenų tipo konstantos skaitmeninę reikšmę, 16 lentelė	

16 lentelė. Palyginimo konstantos

Konstanta	Reikšmė	Kintamojo duomenų tipas
vbEmpty	0	Nenustatytas duomenų tipas
vbNull	1	Negalimas duomenų tipas
vbInteger	2	*Integer*
vbLong	3	*Long integer*
vbSingle	4	*Single*
vbDouble	5	*Double*
vbCurrency	6	*Currency*
vbDate	7	*Date*
vbString	8	*String*
vbObject	9	*Object*
vbError	10	*Error*
vbBoolean	11	*Boolean*
vbVariant	12	*Variant*
vbDataObject	13	*Data access object*
vbDecimal	14	*Decimal*
vbByte	17	*Byte*
vbUserDefinedType	36	Variantas vartotojo nurodytiems duomenims
vbArray	8192	Masyvas

17 lentelė. ANSI kodų valdymo funkcijos.

Sintaksė	Apibūdinimas
Asc *(tekstas)*	Pateikia teksto pirmojo simbolio ANSI kodą, 18 lentelė. Pvz.: *Asc ("A")* ' pateikia 65 *Asc ("ABC")* ' pateikia 65 *Asc ("abc")* ' pateikia 97
Chr *(ansi kodas)*	Pateikia simbolį pagal ANSI kodą, 17 lentelė. Jeigu tai klavišas, tai atlieka jo veiksmą. Pvz.: *Chr (65)* ' pateikia "A" *Chr (97)* ' pateikia "a" *Chr (120)* ' pateikia "x"

18 lentelė. ANSI kodai (0 – 127) ir simboliai

0		**32**	*[space]*	**64**	@	**96**	`
1		**33**	!	**65**	A	**97**	a
2		**34**	"	**66**	B	**98**	b
3		**35**	#	**67**	C	**99**	c
4		**36**	$	**68**	D	**100**	d
5		**37**	%	**69**	E	**101**	e
6		**38**	&	**70**	F	**102**	f
7		**39**	'	**71**	G	**103**	g
8	*[backspace]*	**40**	(	**72**	H	**104**	h
9	*[tab]*	**41**	)	**73**	I	**105**	i
10	*[linefeed]*	**42**	*	**74**	J	**106**	j
11		**43**	+	**75**	K	**107**	k
12		**44**	,	**76**	L	**108**	l
13	*[Enter]*	**45**	-	**77**	M	**109**	m
14		**46**	.	**78**	N	**110**	n
15		**47**	/	**79**	O	**111**	o
16		**48**	0	**80**	P	**112**	p
17		**49**	1	**81**	Q	**113**	q
18		**50**	2	**82**	R	**114**	r
19		**51**	3	**83**	S	**115**	s
20		**52**	4	**84**	T	**116**	t
21		**53**	5	**85**	U	**117**	u
22		**54**	6	**86**	V	**118**	v
23		**55**	7	**87**	W	**119**	w
24		**56**	8	**88**	X	**120**	x
25		**57**	9	**89**	Y	**121**	y
26		**58**	:	**90**	Z	**122**	z
27		**59**	;	**91**	[	**123**	{
28		**60**	<	**92**	\	**124**	|
29		**61**	=	**93**	]	**125**	}
30		**62**	>	**94**	^	**126**	~
31		**63**	?	**95**	_	**127**	

19 lentelė. Laiko skirtumų parametrai

Parametras	Apibūdinimas
yyyy	Metai
q	Ketvirtis
m	Mėnuo
y	Metų diena
d	Diena
w	Savaitės diena
ww	Savaitė
h	Valanda
n	Minutė
s	Sekundė

20 lentelė. Datos ir laiko funkcijos

Sintaksė	Apibūdinimas	Argumentas
Date	Pateikia sistemos laiką mm – nuo 01 iki 12 mėnesio, dd – nuo 1 iki 31 dienos, yy – nuo 1980 iki 2099 metų	
DateAdd *(laikotarpis, skaičius, data)*	Datos pakeitimas nurodytu laikotarpiu, padaugintu iš skaičiaus	Čia data – leistina datos išraiška
DateDiff *(laikas, data1, data2 [, psd [, psm]])*	Suranda laiko skirtumą tarp datų: *data2* ir *data1*. Laiko skirtumas matuojamas *stringais*, laikas – parametrais, 19 lentelė. Kintamieji psd – pirmoji savaitės diena, psm – pirmoji metų diena, 19 lentelė	Čia data – leistina datos išraiška
DatePart *(laikas, data [, psd [, psm]])*	Suranda laikotarpį, kuriam priklauso data. Laikotarpis nurodomas laiko skirtumų parametrais, 19 lentelė. Kintamieji psd – pirmoji savaitės diena, psm – pirmoji metų diena, 19 lentelė	Čia data – leistina datos išraiška
DateSerial *(metai, mėnuo, diena)*	Pateikia datą pagal pateiktus argumentus	Metai nuo 0 ÷ 9999, mėnuo nuo 1 iki 12, diena nuo 1 iki 31
DateValue *(data)*	Konvertuoja tekstą į datos formatą	Data – stringas
Day *(skaičius)*	Pateikia dienos numerį per mėnesį nuo 1 iki 31	Skaičius – bet kokia skaitmeninė išraiška
Hour *(skaičius)*	Pateikia valandos sveikąjį skaičių per parą nuo 0 iki 23	Skaičius – bet kokia skaitmeninė išraiška
Minute *(skaičius)*	Pateikia minutės sveikąjį skaičių per valandą nuo 0 iki 59	Skaičius – bet kokia skaitmeninė išraiška
Month *(skaičius)*	Pateikia mėnesio sveikąjį skaičių per metus nuo 1 iki 12	Skaičius – bet kokia skaitmeninė išraiška
Second *(skaičius)*	Pateikia sekundės sveikąjį skaičių per minutę nuo 0 iki 59	Skaičius – bet kokia skaitmeninė išraiška
Time *[()]*	Pateikia sisteminio laiko reikšmę nuo 0:00:00 iki 23:59:59	
Timer	Pateikia laiko reikšmę sekundėmis	
TimeSerial *(valanda, minutė, sekundė)*	Pateikia laiko reikšmę	Valanda nuo 0 ÷ 23, minutė nuo 0 iki 59, sekundė nuo 0 iki 59
TimeValue *(valanda)*	Pateikia laiko reikšmę	Nuo 0:00:00 (12:00:00 AM) iki 23:59:59 (11:59:59 PM)
Wekday *(data [,pirmoji])*	Pateikia savaitės dienos numerį nuo 1 iki 7	Pirmoji savaitės diena sekmadienis (įvestas) ar pirmadienis (*vbMonday*)
Year *(data)*	Pateikia metų skaitmeninę reikšmę intervale 100 ÷ 9999	Data – leistina datos išraiška

21 lentelė. Datos valdymo konstantos reikšmės

Konstanta	Reikšmė	Apibūdinimas
vbSunday	1	Sekmadienis (nurodyta)
vbMonday	2	Pirmadienis
vbTuesday	3	Antradienis
vbWednesday	4	Trečiadienis
vbThursday	5	Ketvirtadienis
vbFriday	6	Penktadienis
vbSaturday	7	Šeštadienis
VbFirstJan1	1	Skaičiuojama nuo sausio 1 dienos (nurodyta)
vbFirstFourDays	2	Pirmoji skaičiuojamoji savaitė turi 5 dienas
vbFirstFullWeek	3	Pirmoji skaičiuojamoji savaitė turi 7 dienas

22 lentelė. Informacijos pateikimo ir įvedimo funkcijos

Sintaksė	Apibūdinimas
MsgBox *(tekstas [, mygtukai] [, pav])*	Argumento tekstas pateikia informaciją pranešimo lange. Argumentas tekstas yra *stringas* iki 1024 simbolių, kurią galima padalyti į skirtingas eilutes su funkcija *Chr(13)* arba *Chr(10)* ir su šių funkcijų deriniu *(Chr(13) & Chr(10))*. Argumentas *mygtukai* yra skaitmeninės reikšmės (23 lentelė), kurios nurodo pranešimo lango mygtukų tipus. Argumentas *pav* yra pranešimo lango pavadinimas
InputBox *(tekstas [, pav] [, pradinis] [, x] [, y])*	Informaciją tekstas teikia iš pranešimo lango. Argumentai tekstas ir pavadinimas (pav.) yra analogiški kaip *MsgBox* funkcijoje. Argumentas pradinis yra argumento tekstas dažniausiai pasikartojanti reikšmė, kurią galima nuolat matyti įvedimo pozicijoje. Argumentai x ir y nurodo įvedimo lango kairiojo viršutinio kampo horizontalųjį ir vertikalųjį atstumus nuo monitoriaus kairiojo viršutinio kampo

23 lentelė. Pranešimo lango mygtukų tipų konstantos

Konstanta	Reikšmė	Apibūdinimas
vbOKOnly	0	Rodo tik ***OK*** mygtuką
vbOKCancel	1	Rodo ***OK*** ir ***Cancel*** mygtukus
vbAbortRetryIgnore	2	Rodo ***Abort***, ***Retry*** ir ***Ignore*** mygtukus
vbYesNoCancel	3	Rodo ***Yes***, ***No***, ir ***Cancel*** mygtukus
vbYesNo	4	Rodo ***Yes*** ir ***No*** mygtukus

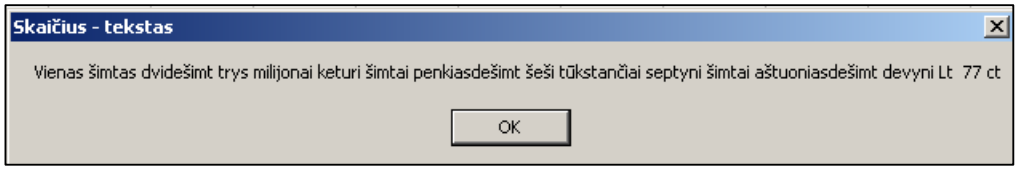
Pavyzdys. Vartotojo funkcijos naudojimas procedūroje.

Aktualu užrašytą pinigų sumą skaičiumi dar parašyti ir tekstu. Tai atlieka procedūra ir dvi vartotojo parašytos funkcijos. Pirmoji funkcija ***SkaiciusTekstas*** sveikuosius skaičius paverčia tekstu, o antroji funkcija ***LitaiCentai*** iškviečia pirmąją funkciją ir gautąjį tekstą įvardija litais bei prideda skaitmeninę centų reikšmę ir pažymi.

```
Public Sub Procedura()
Dim skaicius As Double
Dim stringas As String
skaicius = InputBox("Įveskite skaičių ", "SKAIČIUS-TEKSTAS", 2)
stringas = LitaiCentai(skaicius)
MsgBox stringas, 0, "Skaičius – tekstas"
End Sub
```

33 pav. Skaičiaus įvedimo langas

```
Function LitaiCentai(sk2 As Double) As String
Dim litai As String
Dim centai As String
Dim lt As Long
Dim ilgis As Long

lt = Int(sk2)
centai = (sk2 - CDec(lt)) * 100
litai = SkaiciusTekstas(lt)
ilgis = Len(litai)
LitaiCentai = StrConv(Left(litai, 1), vbUpperCase) & Right(litai, ilgis - 1) & " Lt  " _
& centai & " ct"
End Function
```



34 pav. Programos rezultatas

```
Function SkaiciusTekstas(sk As Long) As String
    Dim dd As Long
    Dim eil As String
      If sk > 19 Then dd = 10
      If sk > 99 Then dd = 100
      Select Case sk
            Case 0: SkaiciusTekstas = "0 "
            Case 1: SkaiciusTekstas = "vienas"
            Case 2: SkaiciusTekstas = "du"
            Case 3: SkaiciusTekstas = "trys"
            Case 4: SkaiciusTekstas = "keturi"
            Case 5: SkaiciusTekstas = "penki"
            Case 6: SkaiciusTekstas = "šeši"
            Case 7: SkaiciusTekstas = "septyni"
            Case 8: SkaiciusTekstas = "aštuoni"
```

```
                Case 9: SkaiciusTekstas = "devyni"
                Case 10: SkaiciusTekstas = "dešimt"
                Case 11: SkaiciusTekstas = "vienuolika"
                Case 12: SkaiciusTekstas = "dvylika"
                Case 13: SkaiciusTekstas = "trylika"
                Case 14: SkaiciusTekstas = "keturiolika"
                Case 15: SkaiciusTekstas = "penkiolika"
                Case 16: SkaiciusTekstas = "šešiolika"
                Case 17: SkaiciusTekstas = "septyniolika"
                Case 18: SkaiciusTekstas = "aštuoniolika"
                Case 19: SkaiciusTekstas = "devyniolika"
                Case Is < 30: eil = "dvidešimt"
                Case Is < 40: eil = "trisdešimt"
                Case Is < 50: eil = "keturiasdešimt"
                Case Is < 60: eil = "penkiasdešimt"
                Case Is < 70: eil = "šešiasdešimt"
                Case Is < 80: eil = "septyniasdešimt"
                Case Is < 90: eil = "aštuoniasdešimt"
                Case Is < 100: eil = "devyniasdešimt"
                Case Is < 200: eil = "vienas šimtas"
                Case Is < 1000: eil = SkaiciusTekstas(Int(sk / 100)) + " šimtai"
                Case Is < 1000000
              eil = SkaiciusTekstas(Int(sk / 1000))
        Select Case Right$(eil, 4)  ' $-stringas
               Case "enas": eil = eil + " tūkstantis"
                Case "šimt", "lika", "mtas", "mtai": eil = eil + " tūkstančių"
               Case Else: eil = eil + " tūkstančiai"
        End Select

  dd = 1000

        Case Else
               eil = SkaiciusTekstas(Int(sk / 1000000))
        Select Case Right$(eil, 4)
                     Case "enas": eil = eil + " milijonas"
                     Case "šimt", "lika", "mtas", "mtai": eil = eil + " milijonų"
                     Case Else: eil = eil + " milijonai"
        End Select

   dd = 1000000

End Select

   If dd <> 0 Then
      sk = Int(sk / dd) * dd – sk        ' Pvz:  (25/10)*10-25= -5
      If sk <> 0 Then eil = eil + " " + SkaiciusTekstas(-sk)
      SkaiciusTekstas = eil
   End If
```

End Function

8. Klasės

Objektiškai orientuotas programavimas – tai programavimo metodologija, pagrįsta programinių objektų visuma, kur kiekvienas iš jų yra tam tikros klasės realizacija. Klasė nurodo objekto struktūrą.

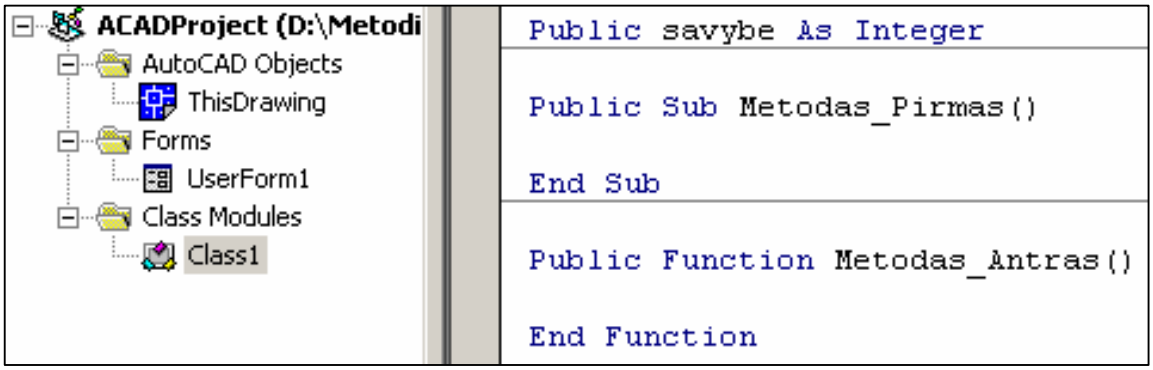
Objektai yra programos (pavyzdžiui, *Excell, Access, Word, Notepad*), valdymo elementai (*CommandButton, TextBox*), programos sukurtos klasės modulyje.

Objektai sąveikauja programoje per metodus, savybes ir įvykius. Metodais programa veikia objektą ir verčia jį reaguoti. Savybės keičia objekto atributus. Įvykiai yra atsakomoji objekto reakcija.

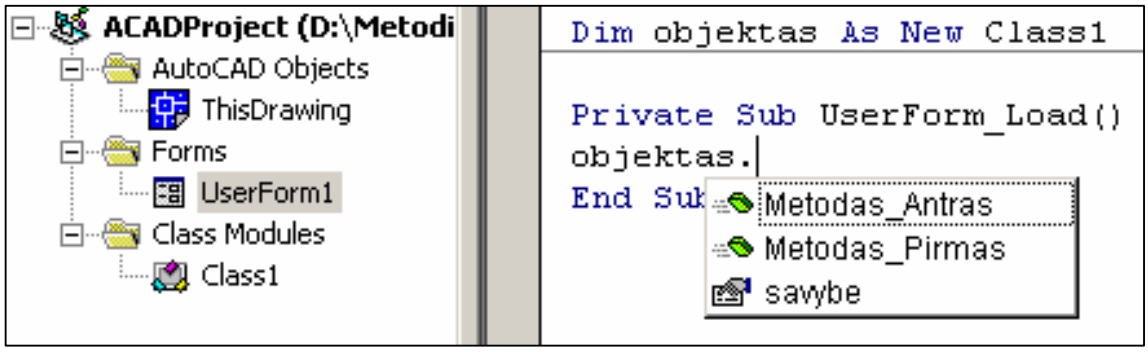
Klasės objektai, jų metodai, savybės ir įvykiai yra programuojami klasės modulyje.

Metodai naudojami objekto funkcinėms galimybėms išplėsti. Metodai skelbiami kaip paprogramės klasės modulyje ***Public Sub*** arba funkcijos ***Public Function*.** Savybės skelbiamos kaip globalieji kintamieji klasės modulyje.

Pavyzdžiui, 35 pav. klasės modulyje paskelbta viena savybė ir du metodai. Kitame, 36 pav., deklaruojamas objektinis kintamasis ir nurodoma, kokiai klasei jis priklauso. Procedūroje panaudoję objektinį kintamąjį ir parašę tašką, galime pasirinkti to objekto metodus ir savybes, nors jos dar nėra aktyvios.



35 pav. Klasės modulis su dviem metodais ir viena savybe



36 pav. Formos modulis su klasės objektu

Pavyzdys. Stačiakampio ir kvadrato ploto skaičiavimas.

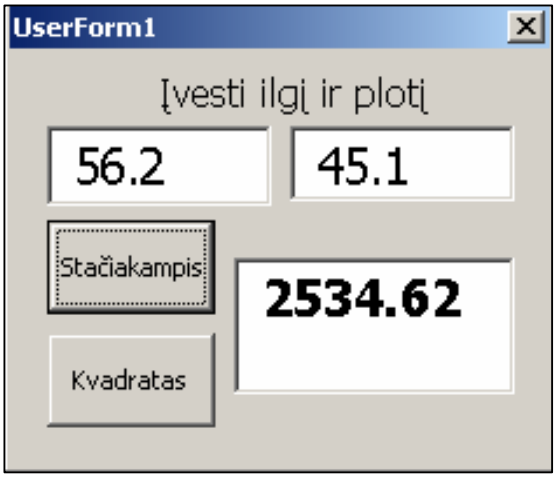
Klasės modulis
Public plotas As Double ' savybė

Public Sub Ploto_sk(pl, il) *'* procedūra su argumentais
*plotas = pl * il*
End Sub

Formos modulyje:
Dim staciak As New Class1
Dim kvadrat As New Class1

Private Sub CommandButton1_Click()
staciak.Ploto_sk TextBox1.Text, TextBox2.Text
TextBox3.Text = staciak.plotas
End Sub

Private Sub CommandButton2_Click()
kvadrat.Ploto_sk TextBox1.Text, TextBox1.Text
TextBox3.Text = kvadrat.plotas
Staciak.
End Sub



37 pav. Vartotojo forma su duomenimis ir rezultatu

Klasės modulyje objekto savybės yra statinės ir dinaminės. Anksčiau naudojome statines savybes, nes jų reikšmės įrašomos ir skaitomos tokios pat, nekeičiant. Statinės savybės deklaruojamos kaip klasės globalieji kintamieji. Jeigu savybės reikšmė kinta, tai tokia savybė yra dinaminė ir aprašoma paprogramiais.

Savybės reikšmei skaityti naudojama paprogramė:

*Public **Property Get** savybės_vardas() As duomenų tipas*
.....
End Property

Savybės reikšmei įrašyti naudojama paprogramė:

*Public **Property Let** savybės_vardas ([argumentai])*
.....
End Property

Klasių savybės gali turėti kelias fiksuotąsias reikšmes. Tam naudojame struktūrą ***Select Case***.

Pavyzdys:

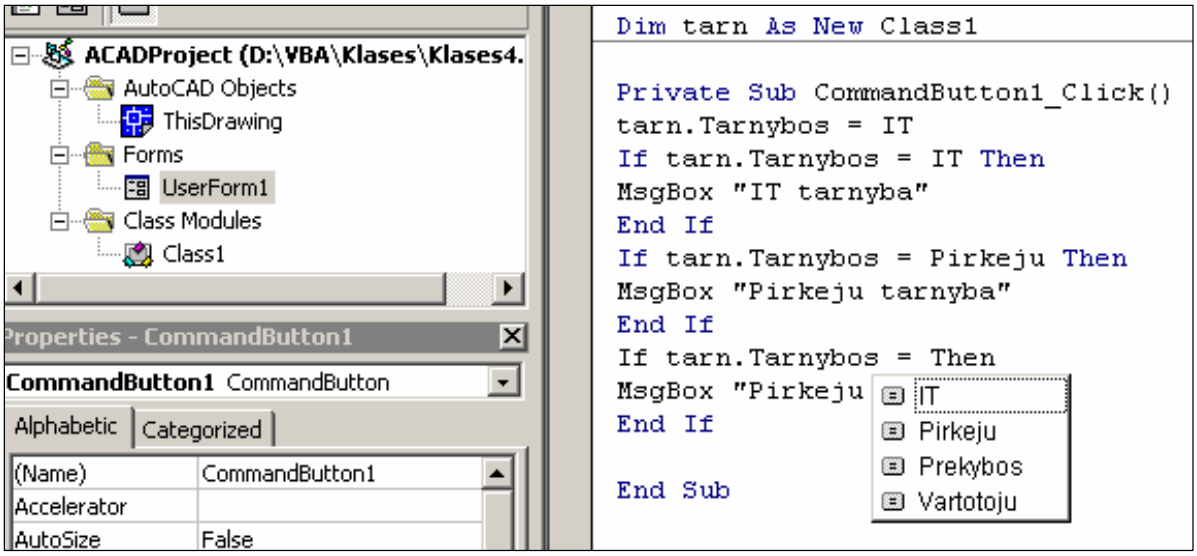
Klasės modulyje:
Public Snr As Integer ' savybe
Public Property Get Spa() As String
Select Case Snr
Case 1: Spa = acGreen
Case 2: Spa = acRed
Case 3: Spa = acYellow
Case 4: Spa = acBlue
End Select
End Property

Public Property Let Spa(ByVal NewVal As String)
Select Case Trim$(UCase$(NewVal))
Case acGreen: Snr = 1
Case acRed: Snr = 2
Case acYellow: Snr = 3
Case acBlue: Snr = 4

```
  End Select
End Property

Formos modulyje:
Dim Sav As New Class2

Private Sub CommandButton1_Click()
Me.Hide
Sav.Snr = TextBox1.Value
Apskritimas
apsk.Color = Sav.Spa
apsk.Update
Me.Show
End Sub
```

Klasių savybės gali turėti kelias fiksuotąsias reikšmes. Tam naudojame operatorių ***Enum***.

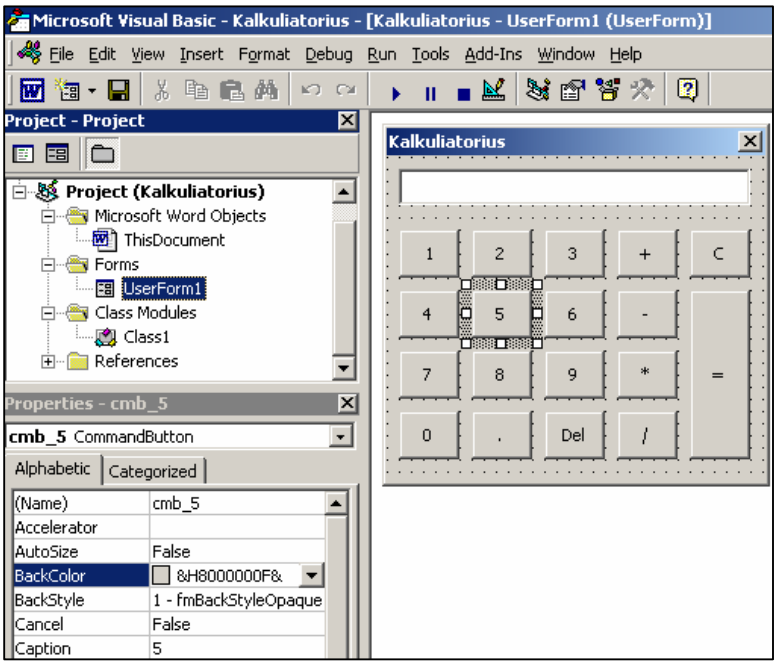
```
Public Enum Tarnyba ' savybė
Prekybos = 101
Vartotoju = 102
Pirkeju = 103
IT = 104
End Enum

Dim TarnNum As Tarnyba

Public Property Get Tarnybos() As Tarnyba
   Tarnybos = TarnNum
End Property

Public Property Let Tarnybos(ByVal NewTarn As Tarnyba)
   TarnNum = NewTarn
End Property
```



38 pav. Antrojo pavyzdžio rezultatai

Pavyzdys kalkuliatorius

Parašykime programą kalkuliatoriui, kuris atliktų keturis aritmetinius veiksmus su realiaisiais skaičiais. Pirmiausia modeliuojame vartotojo formą su visais kalkuliatoriaus mygtukais ir įvedimo-išvedimo langu (39 pav.)



39 pav. Kalkuliatoriaus grafinis modelis

Klasės modulyje deklaruojame pagrindinius kintamuosius. Realiųjų skaičių kintamieji *Sk1* ir *Sk2* yra suformuojami spaudžiant skaitmeninius ir taško mygtukus. Loginis kintamasis *Kitas* turi tik galimas dvi reikšmes: *True*, kada bus atliekamas aritmetinis veiksmas ir įvedamas kitas skaičius; *False*, kada skaičiuosime galutinį rezultatą. Teksto kintamasis *Veiksmas* lygus vienam iš keturių aritmetinių ženklų.

```
Option Explicit
Dim Sk1, Sk2 As Variant
Dim Kitas As Boolean
Dim Veiksmas As String
```

Suteikiame kintamiesiems pradines reikšmes:

```
Private Sub Class_Initialize()
    Sk1 = 0 : Sk2 = 0
    Kitas = False : Veiksmas = ""
End Sub
```

Sukuriame nemažai savybių. Skaičių rašymo savybės:

```
Public Property Let SavybeSk1(ByVal Naujas As String)
    Sk1 = Val(Naujas)
End Property

Public Property Let SavybeSk2(ByVal Naujas As String)
    Sk2 = Val(Naujas)
End Property
```

Veiksmų nuoseklumo sekimo savybė, kuri bus skaitoma ir rašoma:

```
Public Property Get SavybeSeka() As Boolean
    SavybeSeka = Kitas
End Property

Public Property Let SavybeSeka(ByVal Naujas As Boolean)
    Kitas = Naujas
End Property
```

Aritmetinių veiksmų savybė, kuri bus skaitoma ir rašoma:

```
Public Property Get SavybeVeik() As String
    SavybeVeik = Veiksmas
End Property

Public Property Let SavybeVeik(ByVal Naujas As String)
    Veiksmas = Naujas
End Property
```

Klasėje bus vienas metodas, aritmetinių veiksmų atlikimo funkcija:

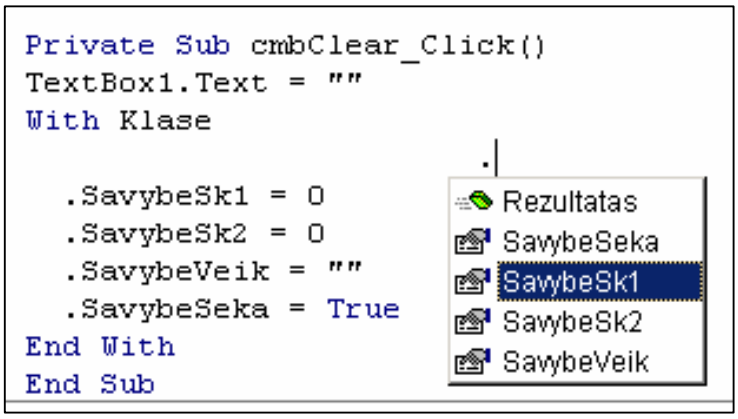
```
Public Function Rezultatas() As String
    Dim r As Double
    r = 0
    Select Case Veiksmas
        Case "+"
        r = Sk1 + Sk2
        Case "-"
        r = Sk1 – Sk2
        Case "*"
        r = Sk1 * Sk2
        Case "/"
        r = Sk1 / Sk2
        Case Else
        r = 0
    End Select
    Rezultatas = Str(r)
End Function
```

Formos modulyje deklaruojame kintamuosius ir naują klasę. Ji turės visas savybes ir metodą, kuriuos pateikėme anksčiau.

```
Option Explicit
Dim Klase As New Class1
Dim s As String
Dim Veiksmas As String
```

Kartu su formos pasirodymu sukuriama ir nauja klasė, kuri turi metodą ir savybes (40 pav.):

```
Private Sub UserForm1_Load()
    Set Klase = New Class1
End Sub
```



40 pav. Klasės metodas ir savybės pasirodo parašius tašką

Procedūra, kuri sukuria skaitmenį įvedimo-išvedimo lange *TextBox1*. Mes spausime mygtukus su skaičiais, o jie atsiras jau esamų skaičių dešinėje.

```
Private Sub Pap_Skaicius(Skaicius As Integer)
    If Klase.SavybeSeka Then
        TextBox1.Text = ""
        Klase.SavybeSeka = False
    End If
    TextBox1.Text = TextBox1.Text & Skaicius
End Sub
```

Skaičių įvedimo mygtuku procedūros:

```
Private Sub cmb_1_Click()
    Pap_Skaicius (1)
End Sub

Private Sub cmb_2_Click()
    Pap_Skaicius (2)
End Sub
……….

Private Sub cmb_9_Click()
    Pap_Skaicius (9)
End Sub

Private Sub cmb_0_Click()
    Pap_Skaicius (0)
End Sub
```

Realiesiems skaičiams įvesti reikalingas taškas, atskiriantis sveikąją skaitmens dalį nuo trupmeninės.

```
Private Sub cmbTaskas_Click()
    If Klase.SavybeSeka Then
        TextBox1.Text = ""
        Klase.SavybeSeka = False
    End If
    If TextBox1.Text = "" Then
```

```
            TextBox1.Text = TextBox1.Text = "0."
        ElseIf InStr(TextBox1.Text, ".") = 0 Then TextBox1.Text = _
            TextBox1.Text & "."
        End If
End Sub
```

Paprogramę *Veiksmas* iškvies matematinių veiksmų mygtukai:

```
Private Sub Pap_Veiksmas()
        With Klase
            .SavybeSk2 = TextBox1.Text
            If .SavybeVeik <> "" Then
                s = .Rezultatas
            Else: s = TextBox1.Text
            End If
            .SavybeSk1 = s
            .SavybeSk2 = "0"
            TextBox1.Text = s
            .SavybeVeik = Veiksmas
            .SavybeSeka = True
        End With
End Sub
```

Aritmetinių veiksmų mygtukai:

```
Private Sub cmbPlius_Click()
        Veiksmas = "+"
        Pap_Veiksmas
End Sub

Private Sub cmbMinus_Click()
        Veiksmas = "–"
        Pap_Veiksmas
End Sub

Private Sub cmbDaug_Click()
        Veiksmas = "*"
        Pap_Veiksmas
End Sub

Private Sub cmbDal_Click()
        Veiksmas = "/"
        Pap_Veiksmas
End Sub
```

Mygtuko *lygu* paspaudimas pateikia galutinį rezultatą.

```
Private Sub cmbEq_Click()
        With Klase
            If .SavybeVeik <> "" Then
                .SavybeSk2 = TextBox1.Text
                TextBox1.Text = .Rezultatas
                .SavybeSk1 = TextBox1.Text
            End If
```

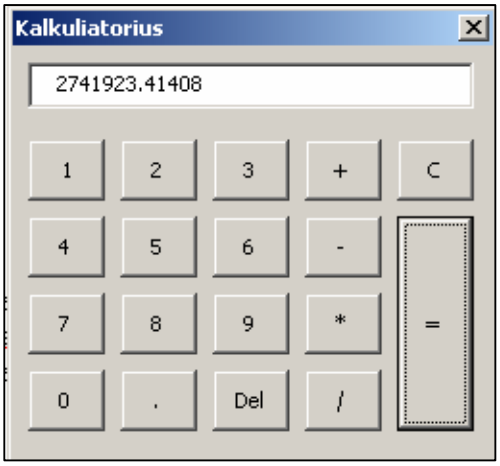
```
            .SavybeSk1 = 0
            .SavybeSk2 = 0
            .SavybeVeik = ""
            .SavybeSeka = True
        End With
End Sub
```

Įvedimo-išvedimo lange rodomas skaičius ištrinamas mygtuku [C]. Šio veiksmo procedūra:

```
Private Sub cmbClear_Click()
        TextBox1.Text = ""
        With Klase
            .SavybeSk1 = 0
            .SavybeSk2 = 0
            .SavybeVeik = ""
            .SavybeSeka = True
        End With
End Sub
```

Įvedimo-išvedimo lange rodomo skaičiaus paskutinis skaitmuo ištrinamas mygtuku [D]. Šio veiksmo procedūra:

```
Private Sub cmbDel_Click()
        s = TextBox1.Text
        TextBox1.Text = Left(s, Len(s) – 1)
End Sub
```



41 pav. Veikiantis kalkuliatorius

Objektas kolekcijos (*collection*) skirtas saugoti duomenims, susietiems pagal objektų prasmę. *Kolekcijos* elementą galima iškviesti pagal indeksą (kaip masyve) ir pagal reikšminį žodį. Objektas *kolekcijos* suprantamas kaip labai lankstus masyvas, kuriame gali būti kiti objektai ir kitos *kolekcijos*. *Kolekcija* skelbiama taip:

Dim *kolekcijos_vardas* ***As New Collection***

Žodis ***New*** reiškia, kad skelbiama įvardyta nauja *kolekcija*. *Kolekcijos* objektas turi keturis metodus pridėti, skaičiuoti, rasti, šalinti (*add, count, item, remove*).

Metodas pridėti (*add*) papildo *kolekciją* nauju elementu:

*kolekcijos_vardas****.Add*** *reikšmė, [raktas], [prieš], [po]*

Argumentas *reikšmė* yra lygi naujam elementui, *raktas* lygus to elemento reikšminiam žodžiui. Argumentai *prieš* ir *po* naudojami atskirai nurodant elemento vietą *kolekcijoje*.

Metodas skaičiuoti (*count*) suranda elementų sumą *kolekcijoje*. Tam naudojamas operatorius:

*Suma = kolekcijos_vardas****.Count***

Metodas rasti (*item*) pateikia konkretų elementą iš *kolekcijos*. Nuoroda į elementą gali būti elemento numeris *kolekcijoje* arba jo reikšminis žodis:

*Objektas = kolekcijos_vardas****.Item*** *(2)*
*Objektas = kolekcijos_vardas****.Item*** *(”Mygtukas2“)*

Standartinį **Item** metodą galima ir praleisti:

Objektas = kolekcijos_vardas (”Mygtukas2“)

Metodas šalinti (*remove*) šalina elementą iš *kolekcijos*. Nuoroda į elementą gali būti elemento numeris (indeksas) *kolekcijoje* arba jo reikšminis žodis:

*kolekcijos_vardas****.Remove*** *(3)*
*kolekcijos_vardas.****Remove*** *(”Mygtukas3“)*

Pridedant arba šalinant elementą *kolekcijoje*, indeksai persinumeruoja automatiškai, todėl programoje jo išsaugoti nereikia.

Pavyzdys. Kolekcijos sudarymas

```
Dim kolekcija As New Collection
Private Sub CommandButton1_Click()
Dim i As Integer
kolekcija.Add "Jonas"
kolekcija.Add "Petras"
kolekcija.Add "Simas"
kolekcija.Add "Valius"
For i = 1 To kolekcija.Count
ListBox1.AddItem kolekcija.Item(i)
Next i
End Sub
```

42 pav. Kolekcija

9. Rinkmenos

Rinkmena (*file*) – tai duomenų rinkiniai, saugomi kompiuterio atminties įrenginiuose. Rinkmenos pavadinimas susideda iš pagrindinės dalies, taško ir iki trijų simbolių rinkmenos tipo. Rinkmenose yra surašytos visos programos ir jų duomenys. Šiame skyriuje nustatysime, kaip galime informaciją programiniu būdu perskaityti ir užrašyti teksto rinkmenoje. Naudosime *Windows* aplinkoje esančio teksto *Notepad* redaktoriaus rinkmenas.

Pagal saugomos informacijos pobūdį galimi nuoseklaus, atsitiktinio ir dvejetainio kreipimosi į rinkmenas tipai. Nuoseklaus kreipimosi būdu skirtingo ilgio informacija įrašoma nuosekliai, be tarpų. Atsitiktinio kreipimosi rinkmenose visa informacija įrašoma vienodo ilgio fragmentais. Dvejetainio kreipimosi rinkmenose informacija įrašoma pagal konkrečios rinkmenos formatą ir galima saugoti bet kokio tipo duomenis. 24 lentelėje yra pateiktos visų tipų rinkmenų valdymo funkcijos.

24 lentelė. Rinkmenų valdymo funkcijos

Sintaksė	Apibūdinimas
Dir *[(kelias [, atributai])]*	Pateikia rinkmenos arba katalogo pavadinimą, kuris atitinka nurodytąją. Pvz.: *Dir ("d:\mano\tekstai")*
EOF *(rinkmenos numeris)*	Pateikia informaciją, kad pasiekta rinkmenos pabaiga
FileDateTime *(kelias)*	Pateikia rinkmenos sukūrimo arba redagavimo datą ir laiką
FileLen *(kelias)*	Pateikia rinkmenos dydį baitais
FreeFile *[(diapazonas)]*	Pateikia laisvą rinkmenos numerį, kurį naudoja operatorius *Open*
GetAttr *(kelias)*	Pateikia skaičių, kuris parodo rinkmenos arba katalogo atributus. 0 – *Normal*, 1 – *Read only*, 2 – *Hidden*, 4 – *System file*, 32 – *Archive*
Input *()*	Pateikia eilutę su visais simboliais, perskaitytais iš rinkmenos atidaryto *Input* arba *Binary* režimais
Kill *(kelias)*	Ištrina nurodytas rinkmenas
Loc *(rinkmenos numeris)*	Pateikia skaičių, kuris atidarytoje rinkmenoje parodo aktyvią skaitymo ir rašymo poziciją
LOF *(rinkmenos numeris)*	Pateikia skaičių, kuris parodo atidarytos rinkmenos dydį baitais
Seek *(rinkmenos numeris)*	Pateikia skaičių, kuris atidarytoje operatoriumi *Open* rinkmenoje parodo aktyvią skaitymo ir rašymo poziciją
SetAttr *kelias, atributai*	Keičia rinkmenų atributus

Toliau nagrinėsime, kaip perskaitome ir užrašome nuoseklaus kreipimosi rinkmenų informaciją (25 lentelė).

25 lentelė. Rinkmenų valdymo operatoriai

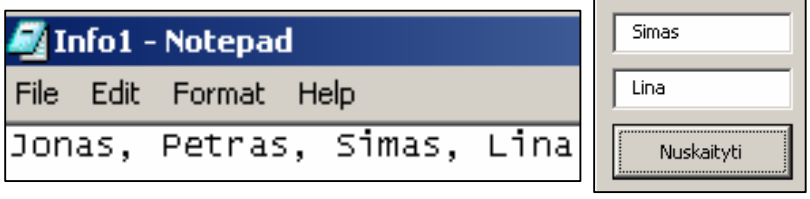
Sintaksė	Apibūdinimas
Close *[[#] rinkmenos numeris %] [,[#] rinkmenos numeris %]...*	Užbaigia operacijas ir uždaro rinkmeną
Input *# rinkmenos numeris %, kintamųjų sąrašas*	Perskaito duomenis iš atidarytos rinkmenos ir priskiria juos kintamiesiems
Line Input *# rinkmenos numeris %, kintamasis*	Perskaito vieną eilutę iš atidarytos rinkmenos ir priskiria ją kintamajam
Open *kelias [**For** režimas] [**Access** operacijos] [leidžiamos operacijos] **As** [#] rinkmenos numeris % [**Len** = įrašo ilgis]*	Atidaro arba sukuria naują rinkmeną. Kiti parametrai 26 lentelėje
Print *# rinkmenos numeris %, [išvedami duomenys]*	Išveda formatuotus duomenis į nurodytą rinkmeną
Write *# rinkmenos numeris %, [išvedami duomenys]*	Išveda duomenis į nurodytą rinkmeną

26 lentelė. Rinkmenų valdymo parametrai

Parametras	Reikšmė	Apibūdinimas
Kelias		*"C:\Darbai\duomenys.txt"* – nurodomas diskas, katalogai ir rinkmena
Režimas	*Append*	Prideda naujus duomenis prie esamų rinkmenoje
	Output	Pakeičia esamus rinkmenoje duomenis naujais. Ankstesni duomenys sunaikinami
	Input	Draudžia įrašyti duomenis į rinkmeną
Operacijos	*Read*	Skaityti duomenis
	Write	Įrašyti duomenis
	Read Write	Skaityti ir įrašyti duomenis
Leidžiamos operacijos	*Shared*	Galima ieškoti informacijos
	Lock Read	Negalima duomenų skaityti
	Lock Write	Negalima duomenų įrašyti
	Lock Read Write	Negalima duomenų skaityti ir įrašyti
Rinkmenos Nr.	Nuo 1 iki 511	Rinkmenos numeris
Įrašo ilgis	<= 32767	Nurodytas skaičius reiškia buferizuojamų simbolių skaičių

Pateikta procedūra nuskaito duomenis iš rinkmenos *Info1.txt*, priskiria kintamiesiems, uždaro rinkmeną ir duomenis parodo forma (43 pav.).

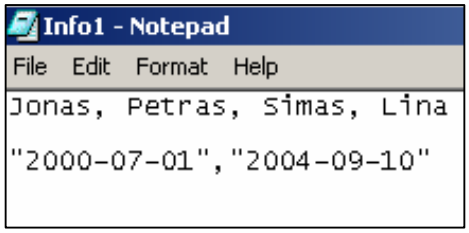
```
Private Sub cmbNuskaityti_Click()
Dim d1, d2, d3, d4, k1, k2 As Variant
Open "D:\Duomenys\Info1.txt" For Input As #1
Input #1, d1, d2, d3, d4
Close #1
TextBox1.Text = d1
TextBox2.Text = d2
TextBox3.Text = d3
TextBox4.Text = d4
End Sub
```



43 pav. Teksto rinkmena ir nuskaityta informacija

Pateikta procedūra priskiria kintamiesiems reikšmes, atidaro rinkmeną *Info1.txt*, prideda prie joje esančių duomenų kintamųjų reikšmes (44 pav.) ir uždaro rinkmeną. Duomenys, pridėti naujoje eilutėje, nes rinkmenoje *Info1.txt* buvo antra tuščia eilutė. Duomenys būtų pridėti toje pačioje eilutėje kaip ir vardai, jeigu žymeklis būtų likęs po paskutinio vardo.

```
Private Sub cmbPridėti_Click()
Dim k1, k2 As Variant
k1 = TextBox5.Text
k2 = TextBox6.Text
Open "D:\Duomenys\Info1.txt" For Append As #2
Write #2, k1, k2
Close #2
End Sub
```

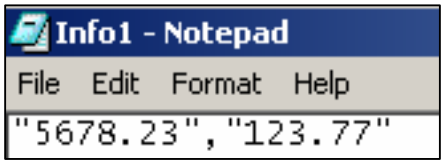


44 pav. Teksto rinkmena su pridėta informacija

Paskutinėje procedūroje pakeitus eilutę ***Open*** tokia:

Open "D:\Duomenys\Info1.txt" For Output As #2

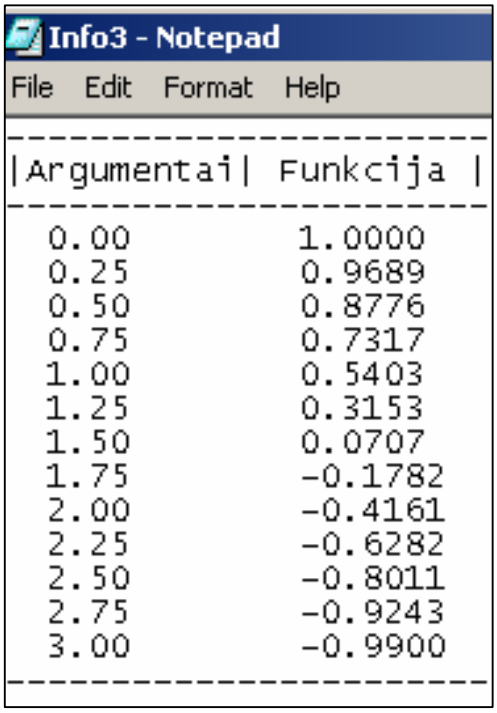
Nauji duomenys pakeistų visą rinkmenoje esančią informaciją (45 pav.)



45 pav. Teksto rinkmena su atnaujinta informacija

Pateikta procedūra atidaro rinkmeną *Info3.txt*, pakeičia joje esančius duomenis funkcijos argumentų ir jos rezultatų lentele (46 pav.) ir uždaro rinkmeną.

```
Private Sub cmbPrint_Click()
Dim x, y As Double
Open "D:\Duomenys\Info3.txt" For Output As #3
Print #3, "-----------------------"
Print #3, "|Argumentai| Funkcija |"
Print #3, "-----------------------"
For x = 0 To 3 Step 0.25
   y = Cos(x)
   Print #3, "  "; Format(x, "#0.00"), Format(x, "#0.0000")
Next x
Print #3, "-----------------------"
Close #3
End Sub
```



46 pav. Rezultatų lentelė teksto rinkmenoje

Literatūra

1. Bayer J. Visual Basic 6. Addison Wesley Professional, 2002. 368 p.
2. Craig J. C., Webb J. Microsoft Visual Basic 6.0. Developer's Workshop. Microsoft Press, 1998. 694 p.
3. Кузьменко В. Г. VBA 2002. Москва: БИНОМ, 2002. 624 с.
4. Eliason A.; Malarkey R. Visual Basic 6.0: Environment, Programming and Applications. Que College Programming, 1999. 784 p.
5. Ostreika A. Programavimo Visual Basic pagrindai. Mokomoji knyga. Kaunas: Technologija, 2003. 225 p.
6. Spasov P. Programming for Technology Students Using Visual Basic. Prentice Hall, 2002. 750 p.
7. Simon R. Practical Visual Basic 6. Que College Programming, 1999. 850 p.
8. Šulcas V. Visual Basic 6 gramatika. Kaunas: „Smaltija", 2003. 160 p.
9. Starkus B. Visual Basic 6 Jūsų kompiuteryje. Kaunas: „Smaltija", 2000. 284 p.

Algirdas Sokas
Programavimas VBA kalba
Mokomoji knyga

Redagavo S. Kirkienė
2005 04 04 6.5 apsk. leid. l.
Leido Vilniaus Gedimino technikos universiteto leidykla „Technika", Saulėtekio al. 11, LT-10223 Vilnius.