ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Sandra Aleksienė

Informatikos specialybės II magistratūros kurso
neakivaizdinio skyriaus studentė

KOMPIUTERINIŲ MATEMATIKOS SISTEMŲ PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBIŲ ANALIZĖ

MAGISTRO DARBAS

Darbo vadovė:
Doc. S. Turskienė

Recenzentė:
Lekt. L. Tankelevičienė

Šiauliai, 2005/2006 m.m.

TURINYS

1. ĮVADAS

Kompiuterinės matematikos sistemos (KMS) atsirado kaip alternatyva tik skaičiuojamąsias galimybes akcentuojančioms universaliosioms programavimo kalboms (UPK). Šiuo metu Lietuvoje populiariausios šios sistemos: *Maple*, *Mathcad*, *Matlab*, *Mathematica*, *Derive* [3].

KMS skirtos matematikos uždaviniams spręsti – jų naudojimo sritys siauresnės, didelis dėmesys skirtas rezultatų vizualumui [1]. Todėl dažniausiai KMS vartojamos kaip analizinių skaičiavimų sistemos. Dar visai neseniai pradėtos tirti KMS programavimo galimybės, [1], [3], [12] darbuose. Šiame darbe bus nagrinėjamos programų grafinės vartotojo sąsajos projektavimo praktinės galimybės.

Tyrimui atlikti bus panaudotos šios KMS: *Matlab* 7, *Maple* 10, *Mathematica* 5.2.

Šio darbo uždaviniai:

1. Išsiaiškinti programų grafinės vartotojo sąsajos kūrimo galimybes nagrinėjamose KMS.
2. Palyginti programų grafinės vartotojo sąsajos kūrimo galimybes KMS, išanalizuoti jų panašumus ir skirtumus.
3. Palyginti KMS ir UPK programų grafinės vartotojo sąsajos kūrimo galimybes, aprašyti jų panašumus ir skirtumus.
4. Sukurti KMS programų pavyzdžių, iliustruojančių komponentų panaudojimą bei kodo rašymo ypatumus.
5. Išsamiai pagrįsti, kuri KMS geriausiai tinka grafinei vartotojo sąsajai projektuoti.

Tikslas – išanalizuoti KMS grafinės vartotojo sąsajos kūrimo galimybes.

Svarbiausias teorinis tyrimo metodas bus palyginimas.

Laukiami rezultatai. Darbe bus išanalizuotos KMS programų grafinės vartotojo sąsajos kūrimo galimybės. Taip pat bus išnagrinėta, ar būtų galima panaudoti KMS taikomosioms programoms kurti ir programavimui mokytis.

2. TEORINĖ DALIS

2.1. PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBĖS

Pastaruoju metu įvairiose programavimo kalbose didelis dėmesys skiriamas grafinės vartotojo sąsajos kūrimo priemonėms. Dar daugiau – netgi teigiama, kad programų komercinę sėkmę 90% lemia jos išorinė sąsaja ir tik apie 10% - jos funkcionalumas [10]. Todėl programuotojui būtina mokėti ne tik rašyti algoritmus, bet ir kurti efektyvias vartotojo sąsajas.

Grafinė vartotojo sąsaja kuriama dviem pagrindiniais būdais:

- ✓ programavimo kalba aprašant sąsajos elementus, jų savybes ir reakciją į įvykius;
- ✓ komponentus perkeliant iš vaizdinių komponentų rinkinių (palečių) į projekto šabloną, komponento savybes nustatant tam tikrose dialogo langų lentelėse ir programuojant tik reakciją į įvykius.

Čia reakcija į įvykius suprantama kaip reakcija į aktyviems komponentams perduotus duomenis ir įvairius vartotojo veiksmus.

Šiuo metu sparčiai plinta antrasis grafinės vartotojo sąsajos kūrimo būdas. Programos sudaromos komponuojant jas iš dalių, derinant, nustatant tarpusavio ryšius. Programuotojui nedaug bereikia rašyti pačiam – sistema pateikia daug įvairių dalių, lieka tik jas sujungti. Komponentų naudojimas ir derinimas – svarbi grafinės vartotojo sąsajos projektavimo dalis.

2.2. PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBĖS MATLAB 7 TERPĖJE

Remiantis [2] straipsniu, *Matlab* 5 versijoje grafinė vartotojo sąsaja buvo kuriama *m*-rinkmenoje iš specialių komandų ir funkcijų (*uitools*). 2002 m. *Matlab* 6.5 versijoje jau palaikomas vaizdinis programavimas.

Sistemoje *Matlab* 7 programų grafinė vartotojo sąsaja kuriama dviem būdais:

- ✓ Rašant programos kodą sistemos *Matlab* darbalaukyje.
- ✓ Naudojant grafinės vartotojo sąsajos kūrimo terpę GUIDE: iš komponentų palečių nešant reikiamus komponentus į specialią sritį *Layout Editor* – taip suformuojamas grafinis vaizdas, o programuojant tik reakciją į įvykius.

Šiuo metu populiaresnis pastarasis grafinės vartoto sąsajos kūrimo būdas, todėl toliau bus kalbama tik apie grafinės vartotojo sąsajos kūrimą vaizdžiuoju būdu.

Kalbant apie programų grafinės vartotojo sąsajos kūrimą *Matlab* terpėje, reiktų išskirti dvi esmines sąvokas:

- ✓ GUIDE – grafinės vartotojo aplinkos kūrimo terpė (angl. Graphical User Interface Development Environment). Šioje terpėje yra įrankių rinkinys, leidžiantis sukurti patogią ir lengvai valdomą grafinę aplinką.
- ✓ GUI – grafinė vartotojo sąsaja (angl. Graphical User Interface). Gerai sukurta programos GUI gali labai palengvinti programos vartojimą. Programa valdoma intuityviai išdėstant mygtukus, sąrašo komponentus, meniu ir kitus komponentus. Programos GUI turi būti sukurta taip, kad programos vartotojas žinotų, ko tikėtis atlikus vieną ar kitą veiksmą [4].

2.2.1. GRAFINĖS VARTOTOJO SĄSAJOS PAGRINDINIAI ELEMENTAI SISTEMOJE MATLAB 7

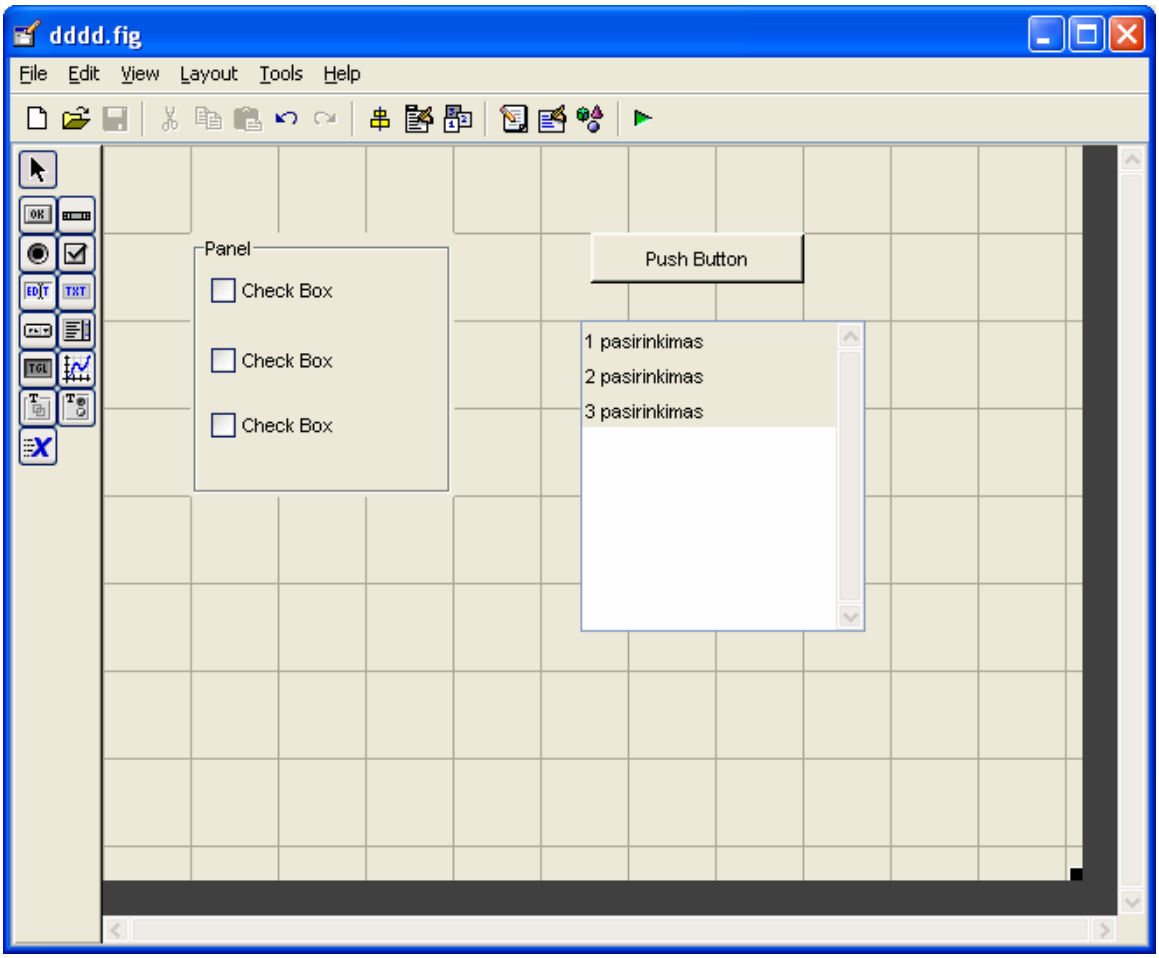
Grafinė vartotojo sąsaja leidžia vartotojui dirbti įprastoje, gerai pažįstamoje terpėje. Ši terpė, sudaryta iš mygtukų, teksto, redagavimo, sąrašo komponentų, meniu, leidžia vartotojui sukoncentruoti dėmesį į programos taikymą, naudojimą, o ne į tai, kaip padarytas vienas ar kitas programos objektas. Tačiau sukurti grafinę vartotojo aplinką programuotojui yra sunkiau, nes šios programos turi būti paruoštos bet kokiam pelės spragtelėjimui ant GUI elemento ar galimam duomenų įvedimui iš klaviatūros. Toks duomenų įvedimas vadinamas įvykiu, o programos, reaguojančios į įvykius, vadinamos į įvykius reaguojančiomis programomis [5].

Norint sukurti grafinę vartotojo sasają *Matlab* 7 sistemoje, reikia trijų pagrindinių elementų:

1. *Komponentų*. Jų yra keletas tipų: grafiniai elementai (mygtukai, redagavimo komponentai, sąrašai), statiniai elementai (rėmeliai, teksto komponentai), meniu ir ašys.
2. *Formos*. GUI komponentai turi būti sutvarkyti formos viduje (forma – tai langas kompiuterio ekrane). Atsidarius programos GUI, sukuriama tuščia forma, kuri gali turėti bet kokias komponentų kombinacijas.
3. *Grįžtamojo ryšio* (angl. callbacks). Pagaliau turi būti būdas atlikti veiksmą, kai vartotojas spragteli pele ar surenka informaciją klaviatūra. Pelės spragtelėjimas yra įvykis – *Matlab* programa turi reaguoti į kiekvieną įvykį. Pavyzdžiui, jei vartotojas paspaudžia mygtuką, šis įvykis turi iškviesti mygtuko įvykdymo funkciją. Šis *Matlab* kalbos kodas, įvykdantis atsaką į įvykį, vadinamas grįžtamuoju ryšiu. Programos GUI turi būti grįžtamasis ryšys iš kiekvieno elemento [4].

2.2.2. MATLAB 7 SISTEMOS PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO ETAPAI

Matlab 7 sistemoje grafinė vartotojo sąsaja kuriama naudojant grafinės vartotojo aplinkos kūrimo terpę, vadinamą GUIDE. Ši terpė leidžia programuotojui sukurti programos GUI, išsirenkant ir išdėstant komponentus formoje. Kai komponentai yra reikiamoje vietoje, programuotojas gali redaguoti jų savybes: vardą, spalvą, dydį, šriftą, rodomą tekstą ir pan. Kai GUI išsaugoma, sukuriama programa, kurioje yra funkcijų skeletai, kuriuos programuotojas gali modifikuoti taip, kad realizuotų grafinės vartotojo sąsajos elgesį [4].

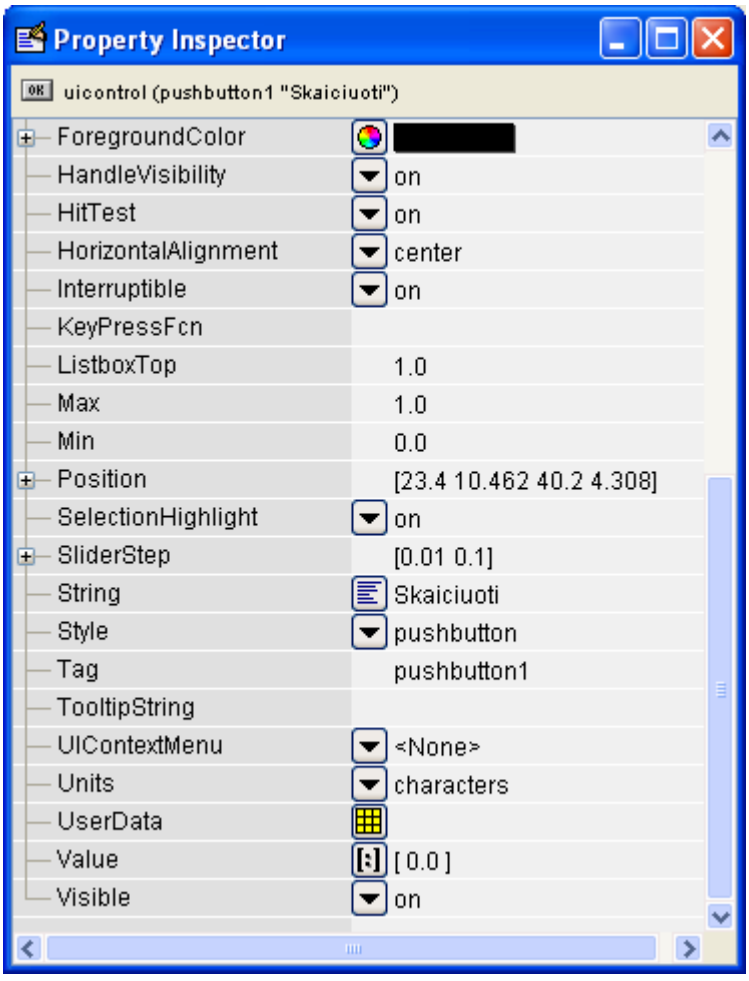


2.1 pav. *Layout Editor* langas

Layout Editor langas turi GUI komponentų paletę (*Component Palete*) kairėje lango pusėje (žr. 2.1 pav). Vartotojas gali kurti GUI komponentus, spragtelėjęs pele ant norimo komponento ir nešdamas jį ant lango. Lango viršuje yra juosta su daugeliu naudingų įrankių, kurie leidžia vartotojui sulygiuoti GUI komponentus, modifikuoti jų savybes, sudaryti meniu ir pan.

Pagrindiniai žingsniai, kuriuos reikia atlikti norint sukurti grafinę vartotojo sasają:

1. Nuspręsti, kokie komponentai bus reikalingi ir kokios bus jų funkcijos.
2. Panaudoti *Matlab* sistemos terpę GUIDE tam, kad išdėstyti komponentus ant formos. Komponentų dydis, išdėstymas ir tarpai tarp komponentų turi būti sureguliuoti, naudojant GUIDE priemones.
3. Panaudoti *Matlab* programos įrankį *Property Inspector* (2.2 paveikslas) tam, kad suteikti kiekvienam komponentui originalų vardą (*tag*) ir nustatyti jo savybes – spalvą, rodomą tekstą ir pan.
4. Išsaugoti formą. Kai forma išsaugoma, yra sukuriamos dvi bylos, kurios turi tuos pačius pavadinimus, bet skirtingus prievardžius. Byloje su prievardžiu *fig* yra GUI komponentai. Byloje su prievardžiu *m* yra kiekvieno komponento grįžtamojo ryšio funkcijų (callback) skeletai.
5. Parašyti programos kodą kiekvienai grįžtamojo ryšio funkcijai, kuri nustatytų komponento elgesį įvykus tam tikram įvykiui.



2.2 pav. Property Inspector langas

2.3. GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBĖS MAPLE 10 TERPĖJE

2002 m. *Maple* sistemoje įdiegta nauja technologija *Maplets*, leidžianti kurti grafinę vartotojo sąsają, rašant programos kodą. 2005 m., pasirodžius *Maple 10* versijai, atsirado galimybė kurti grafinę vartotojo sąsają vaizdžiuoju būdu, naudojantis *Maplet Builder* programų kūrimo terpe.

Maplet taikomoji programa (angl. maplet) – grafinė vartotojo sąsaja, galinti turėti langus, teksto įvedimo laukelius ir kitus vaizdinius komponentus ir įgalinanti plačiau panaudoti sistemos *Maple* galimybes.

Maplet programą galima sukurti dviem būdais:

1. Naudojant *Maplet* programų kūrimo paketą *Maplets*, rašant šablonų rinkinius sistemos *Maple* darbalaukyje.
2. Naudojant programų kūrimo terpę *Maplet Builder*. „Tempiant ir metant“ elementus, nustatant komponentų savybes, veiksmus, susijusius su komponentais, ir tiesiogiai paleidžiant *Maplet* programą.

Maplets paketas turi šiuos vidinius paketus: *Elements*, *Tools*, *Utilities* ir *Examples*. Pakete *Elements* saugomi komponentai, kurie naudojami grafinei vartotojo sąsajai kurti, pvz., mygtukai, teksto laukeliai, išskleidžiami sąrašai, taip pat komponentų savybės, kurios apibrėžia komponentų išsidėstymą ir išvaizdą. Komponentai klasifikuojami į septynias grupes: komandų, dialogo, išdėstymo, meniu, įrankių juostos, langų ir kitų elementų. Paketuose *Tools* ir *Utilities* yra papildomos komandos, skirtos sudėtingesnėms sąsajoms kurti. Pakete *Tools* saugomos komandos, kurios skirtos manipuliuoti ir bendrauti su vartotojo sąsajomis ir vartotojo sąsajų aprašais. Pakete *Utilities* yra pagalbinės komandos, reikalingos grafinei vartotojo sąsajai konstruoti, pvz., dialogo langas klaidoms pranešti. Pakete *Examples* saugomi grafinės vartotojo sąsajos pavyzdžiai, kuriuos galima vartoti vienus ar su kitomis sąsajomis [2].

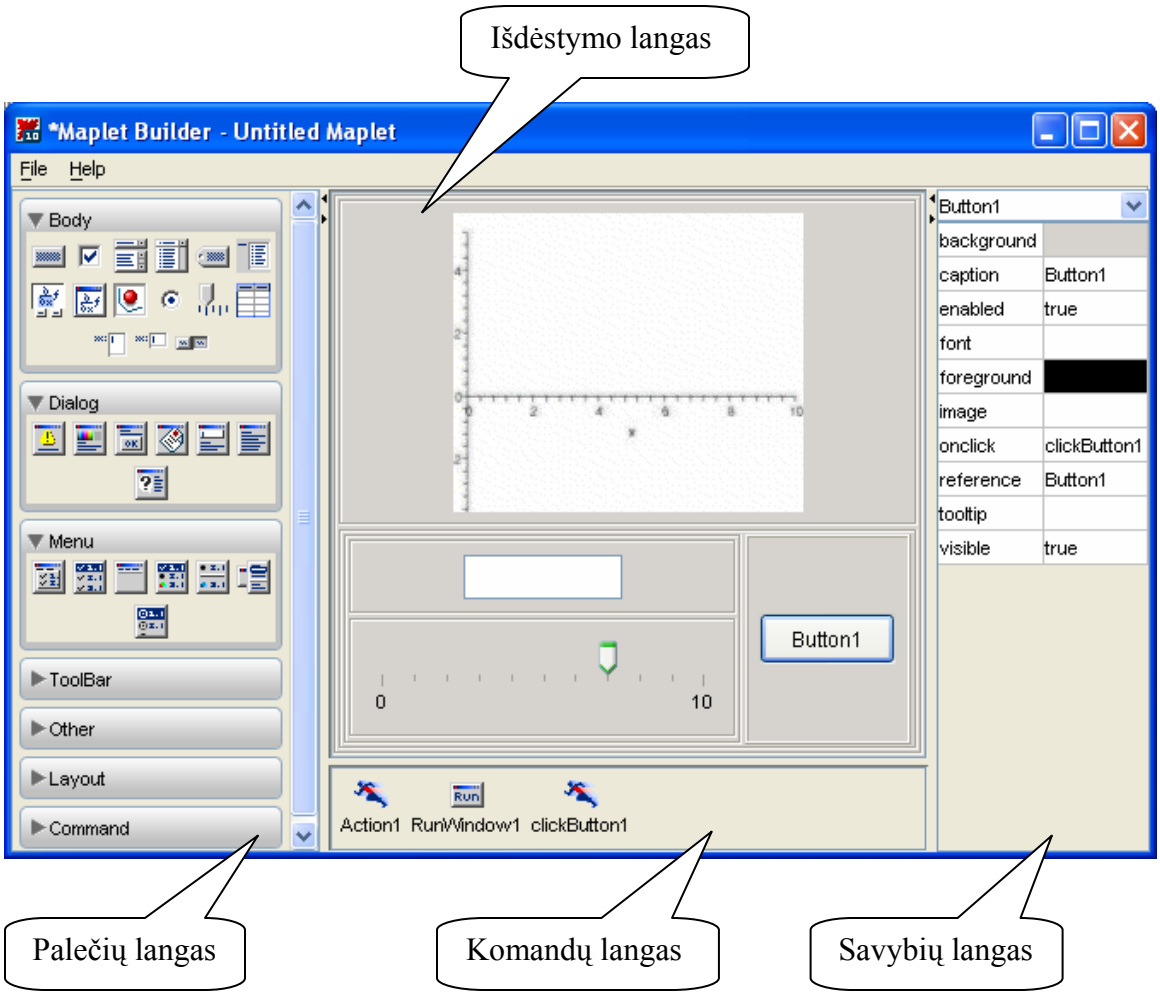
Maplet programos paleidžiamos įvykdžius *Maplet* kalbos kodą. *Maplet* programos kodas gali būti išsaugotas su prievardžiu *.maplet* arba kaip paprastas *Maple* dokumentas su prievardžiu *.mw*. Komanda *Maplet[Display]* įvykdo *Maplet* taikomąją programą, kuri parašyta sistemos *Maple* komandų eilutėje. Vartotojo grafinė sąsaja kuriama įtraukiant elementus į sarašą, kurio aukščiausio lygio elementas yra *Maple( )* elementas.

Maplet Builder yra programų grafinės vartotojo sąsajos kūrimo terpė besisiejanti su *Maplets* paketu. Naudojant *Maplet Builder* galima išdėstyti *Maplet* programos elementus (vaizdinius ir funkcinius *Maplet* programos komponentus), „tempti ir numesti“ (angl. drag and drop) principu, nustatyti veiksmus, susijusius su elementais ir tiesiogiai paleisti *Maplet* programą. *Maplet Builder* paketas yra prieinamas tik iš standartinės komandų eilutės (*Standard Worksheet*) [7].

Patekti į *Maplet Builder* aplinką galima išsirinkus pagrindinio meniu punktus: **Tools -> Assistants -> Maplet Builder**.

Maplet Builder aplinka sudaryta iš keturių langų (žr. 2.3 pav):

1. Palečių langas (*Palette Pane*), sudarytas iš *Maplet* programos komponentų, suskirstytų pagal kategorijas.
2. Išdėstymo langas (*Layout Pane*) rodo komponentus, kurie sudaro konkrečią *Maplet* aplikaciją.
3. Komandų langas (*Command Pane*) rodo komandas ir atsakomuosius veiksmus į įvykius.
4. Savybių langas (*Properties Pane*) rodo konkretaus dabar pažymėto komponento savybes [6].



2.3 pav. *Maplet Builder* langas

Paletės lange yra septynios komponentų palečių grupės. Norint sužinoti komponento vardą, užtenka nuvesti pelę ties komponento ikona. Komponentai iš pagrindinės, įrankių juostos ir išdėstymo palečių tiesiog nešami su pele į norimą vietą išdėstymo lange (*Layout pane*). Komponentai iš dialogo, menu, komandų ir kitų elementų palečių, nešami į komandų langą (*Command pane*). Kai naujas komponentas įdedamas į *Maplet* programą, komandų lange automatiškai įdedami atitinkami veiksmai, susiję su tuo komponentu. Pavyzdžiui, įdėjus mygtuko komponentą išdėstymo lange, automatiškai sukuriamas *clickButton* įvykis. Programuotojui belieka nustatyti, koks konkretus veiksmas bus atliktas paspaudus mygtuką.

Išdėstymo langas (*Layout pane*) yra pagrindinis programų kūrimo terpėje *Maplet Builder*, nes jis rodo konkrečios *Maplet* programos vaizdą. Skirtingai nuo kitų KMS, *Maplet Builder* terpėje elementų išsidėstymą lange nurodo rėmeliai, kurie sukuriami *BoxColumn* ir *BoxRow* elementų kombinacijomis. Eilučių ir stulpelių skaičius nustatomas savybių lange (*Properties pane*).

Spragtelėjus pele komponentą, esantį išdėstymo ar komandų lange, atsiveria tą komponentą atitinkantis savybių langas (*Property pane*).

Komandų lange nurodomi veiksmai, komandos ar meniu elementai, vartojami konkrečioje *Maplet* programoje.

Savybių langas leidžia nustatyti savybes *Maplet* programos komponentams, pavyzdžiui, komponento spalvą, tekstą ar veiksmus, kurie turi įvykti spragtelėjus pele komponentą [7].

2.4. GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBĖS MATHEMATICA 5.2 TERPĖJE

2004 m. lapkričio 17d. pasirodė *Mathematica 5.1* versija, kurios viena iš naujovių – *GUIKit* paketas – suteikia galimybę kurti grafinę vartotojo sąsają, rašant programos kodą.

Sistemoje *Mathematica 5.2* programų grafinę vartotojo sąsają galima kurti tik programuojant komponentus bei veiksmus atliekančius metodus. Komponentų paletės bei galimybės kurti graifinę vartotojo sąsają vaizdžiuoju būdu kol kas dar nėra.

Java kalbos įrankių rinkinys *J/Link*, naudojamas sistemoje *Mathematica 5.2*, suteikia prieigą prie *Java* klasių – tame tarpe prie plačios *Java* klasių bibliotekos grafinei vartotojo sąsajai kurti. *GUIKit* – paketas, suteikiantis galimybę kurti grafinę vartotojo sąsają *J/Link* pagrindu, tačiau grafinė vartotojo sąsaja aprašoma *Mathematica* programavimo kalba. Todėl programuotojas gali nemokėti *Java* programavimo kalbos [9].

Taigi *GUIKit* paketas:

- ✓ suteikia galimybę kurti grafinę vartotojo sąsają, kuri integruojasi su sistema *Mathematica*;
- ✓ leidžia konstruoti komponentus ir jų grupes bei atlikti paprasčiausias užduotis;
- ✓ susisiekia su *Java* klasėmis, naudojantis *J/Link*, tačiau programavimo žinios *Java* aplinkoje nėra reikalingos;
- ✓ veikia daugialypėse terpėse, skirtingose platformose;
- ✓ leidžia sistemoje *Mathematica* atliktiems analizės rezultatams sąveikauti su grafinės vartotojo sąsajos komponentais [8].

Vienas iš didžiausių *GUIKit* privalumų yra greitas grafinės vartotojo sąsajos kūrimo aplinkos išsikvietimas. Prieš naudojant bet kurį *GUIKit* elementą ar funkciją būtina surinkti komandą *Needs[GUIKit]*.

Sukurtai grafinės vartotojo sąsajos aplinkai iškviesti naudojama komanda *GUIRun[pavadinimas]* arba *GUIRunModal[pavadinimas]*.

Grafinės vartotojo sąsajos aplinka kuriama komponentų hierarchijos pagrindu. Aplinka pradedama kurti nuo labiausiai nutolusio komponento, pavyzdžiui, lango (window) ar rėmelio (frame).

Objekto reakcija į įvykį aprašoma tokiu sakiniu:
BindEvent[„įvykio pavadinimas“, Script[išraiška]].

Metodai aprašomi *Script* dalyje sistemos *Mathematica* vidine programavimo kalba.

Skliaustų pagalba nurodomas elementų išdėstymas.

Widget[„komponentoVardas“,{turinys}, pasirinkimai] yra pagrindinis sakinys, naudojamas grafinės vartotojo sąsajos konstravimui. Komponento turinys gali apimti savybių nustatymus, pavyzdžiui, tekstą, rodomą ant komponento. Turinyje taip pat gali būti kiti komponentai ir reakcijos į įvykius kodai [8].

Grafinės vartotojo sąsajos kūrimo *GUIKit* pagalba kūrimo etapai:

- **Komponentų pasirinkimas.** Yra galimybė pasirinkti komponentą iš plačios komponentų kolekcijos. Taip pat galima kurti naujus komponentus.
- **Pradinių reikšmių nustatymas.** Savybės privalo turėti pradines reikšmes. Pavyzdžiui, turi būti nustatytos tokios komponento pradinės savybės kaip tekstas, matomas ant komponento.
- **Komponentų išdėstymas.** Turi būti suprojektuotas grafinės vartotojo sąsajos elementų išdėstymas. Automatinės elementų išdėstymo priemonės šį darbą padaro gana lengvą.
- **Kodo rašymas.** Kodas nusako reakciją į įvairiausius įvykius, kurie gali įvykti vartotojui naudojanti grafine aplinka.

3. PROJEKTINĖ DALIS

3.1. ĮRANKIŲ IR PRIEMONIŲ PASIRINKIMO ANALIZĖ

Tyrimui atlikti pasirinktos šios KMS: *Matlab 7*, *Maple 10*, *Mathematica 5.2*. Šios sistemos pasirinktos dėl kelių priežasčių:

- ✓ Šios KMS populiarios Lietuvos aukštosiose mokyklose.
- ✓ Jos visos turi grafinės vartotojo sąsajos kūrimo galimybes.
- ✓ Šios KMS turi vidines aukšto lygio programavimo kalbas.
- ✓ Šios KMS turi ne tik aukšto lygio matematinį aparatą, bet ir plačias programavimo bei komponentinio programavimo galimybes. O šiuos du dalykus ypač patogu suderinti.

Tam, kad galima būtų palyginti KMS ir UPK programų grafinės vartotojo sąsajos galimybes, parinkta sistema *C++ Builder 6*. Pasirinkimo priežastys:

- ✓ *C++ Builder* sistema plačiai naudojama Lietuvos aukštosiose mokyklose.
- ✓ Taikomųjų programų kūrimas šioje sistemoje paremtas grafinės vartotojo sąsajos kūrimu.
- ✓ *C++ Builder* sistemoje galima realizuoti tiek funkcinę-procedūrinę, tiek ir objektinę programavimo paradigmas.
- ✓ Šia sistema galima sukurti įvairaus lygio ir paskirties programas: tiek mėgėjiškas, tiek profesionalias.

3.2. DARBO VYKDYMO PLANAS

1. Išanalizuoti atskirai kiekvieną KMS ir išsiaiškinti grafinės vartotojo sąsajos kūrimo galimybes.
2. Įsisavinti *Matlab*, *Maple*, *Mathematica* sistemų programavimo konstrukcijas, žinias, kurios bus reikalingos analizei atlikti.
3. Sukurti programas kiekvienoje KMS ir jas palyginti.
4. Išanalizuoti sudėtingesnes grafinės vartotojo sąsajos kūrimo galimybes kiekvienoje KMS.

5. Sukurti programų, realizuojančių klasikinius programavimo uždavinius: panaudoti masyvus, skaitymą iš bylos, įrašus.
6. Sukurti programų, naudojančių įvairesnius grafinės sąsajos elementus, tokius kaip meniu.
7. Išanalizuoti visų KMS grafinės vartotojo sąsajos kūrimo ypatumus jau sukurtose programose.
8. Sukurti tas pačias programas su *C++ Buider* programa.
9. Išanalizuoti programų, sukurtų su KMS, ir programų, sukurtų su *C++ Builder* sistema panašumus ir skirtumus. Taip pat kūrimo eigos ypatumus.
10. Sukurti programų, kurių *C++ Builder* sistemoje vargu, ar pavyktų sukurti.
11. Aprašyti atliktą darbą ir padaryti reikšmingas išvadas.

3.3. PRADINIS PROJEKTO APRAŠYMAS

Tyrimas susideda iš šių etapų:

1. Grafinės vartotojo sąsajos palyginimui, parinkti keturi konkretūs uždaviniai, išspręsti sistemose *Matlab 7*, *Maple 10*, *Mathematica 5.2* ir *C++ Builder 6.*
2. Atliekama konkrečių programų grafinės vartotojo sąsajos analizė tokiais aspektais:
    a) Užduoties analizė.
    b) Programos formos schema.
    c) Komponentų paskirtis.
    d) Savybių lange keičiamos komponentų savybių reikšmės.
    e) Įvykiai, į kuriuos reaguoja programos komponentai.
    f) Programos tekstai.
    g) Programos darbo langas.
    h) Kodo kūrimo ypatumai.
    i) Pastebėjimai.
3. Atliekama pagrindinių komponentų palečių analizė.
4. Analizuojami KMS programų grafinės vartotojo sąsajos panašumai ir skirtumai.
5. Analizuojami KMS ir *C++ Builder* programų grafinės vartotojo sąsajos panašumai ir skirtumai.

4. DARBO EIGOS APRAŠYMAS

4.1. DARBŲ EIGOS GRAFAS

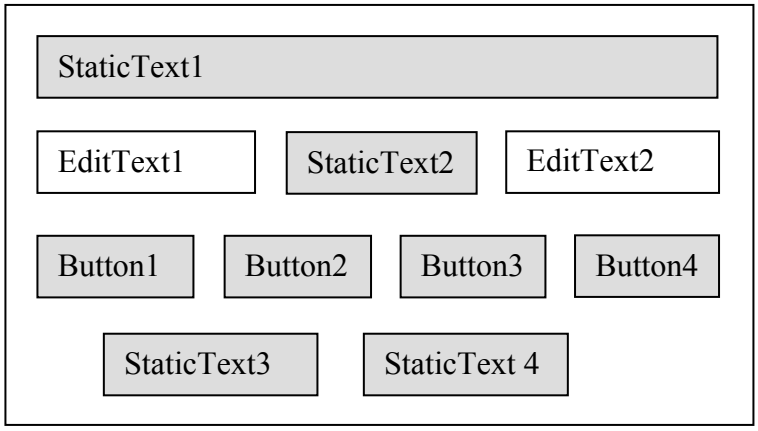
Laikas	**Atlikti darbai**	**Susitikimai su vadove**
I semestras	Įsisavintos kiekvienos KMS programavimo konstrukcijos. Išsiaiškintos kiekvienos KMS grafinės vartotojo sąsajos kūrimo bendriausios galimybės. Sukurtos programos „Skaičiuoklis“ visose KMS ir *C++ Builder* programose.	4 susitikimai
II semestras	Išsamiai išnagrinėtos grafinės vartotojo sąsajos kūrimo galimybės sistemoje *Matlab*. Sukurtos programos „Laipsnis“ ir „Vienmatis masyvas“, kurias vėliau nuspręsta į darbą nebedėti.	4 susitikimai
III semestras	Išsamiai išnagrinėtos grafinės vartotojo sąsajos kūrimo galimybės sistemose *Maple* ir *Mathematica*. Sukurta programa „Studentai“, naudojanti įrašo duomenų tipą sistemoje *Matlab*. Sudaryta pagrindinių komponentų lentelė.	4 susitikimai
IV semestras	Sukurtos programos „Dvimatis masyvas“ visose sistemose. KMS grafinės vartoto sąsajos kūrimo galimybės palyginamos tarpusavyje ir su programos *C++ Builder* galimybėmis. Sukuriamos programos „Diferencijavimas - Integravimas“ nagrinėjamose KMS. Per visus semestrus atliktas darbas sudedamas į vieną ir pridedami nauji pastebėjimai, aprašomi darbo rezultatai.	4 susitikimai

4.2. KONKREČIŲ PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO PROCESO KMS PALYGINIMAS

1 užduotis. *Sukurti programą, atliekančią paprasto skaičiuoklio funkcijas, t.y. jos priemonėmis galima atlikti dviejų klaviatūra įvedamų skaičių sudėtį, atimtį, daugybą ir dalybą.*

Užduoties analizė. Pradiniai duomenys – du realieji skaičiai. Veiksmams atlikti sistemose *Matlab*, *Mathematica* ir *C++ Builder* sukurti keturi metodai, atliekantys aritmetinius veiksmus. Sistemoje *Maple* kiekvieno mygtuko paspaudimui išrenkama OnClick savybė ir užpildoma *Evaluate Expression* kortelė.

Programos formos schema pateikta 4.1 paveiksle.



4.1 pav. Programos „Skaičiuoklis“ formos schema

4.1 lentelė

Komponentų paskirtis

Vardas	Paskirtis
EditText1, EditText2	Duomenų įvedimo langelis
Button1 – Button4	Mygtukai aritmetiniams veiksmams (sudėčiai, atimčiai, daugybai, dalybai) atlikti
StaticText1, StaticText3	Programos paaiškinimų langelis
StaticText 2	Ženklo išvedimo langelis
StaticText 4	Rezultato langelis

4.2 lentelė

Savybių lange keičiamos komponentų savybių reikšmės

Vardas	Keičiama savybė	Reikšmė
EditText1 – 2	String	12, 5
Button1	String	+
Button2	String	-
Button3	String	*
Button4	String	/
StaticText 1	String	Įveskite skaičius ir paspauskite reikiamą mygtuką
StaticText 2	String	?
StaticText 3	String	Rezultatas =
StaticText 4	String	

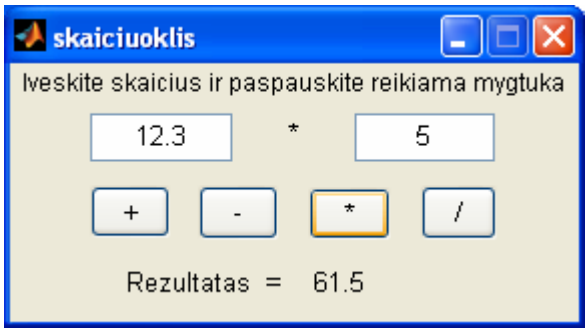
4.3 lentelė

Įvykiai, į kuriuos reaguoja programos komponentai

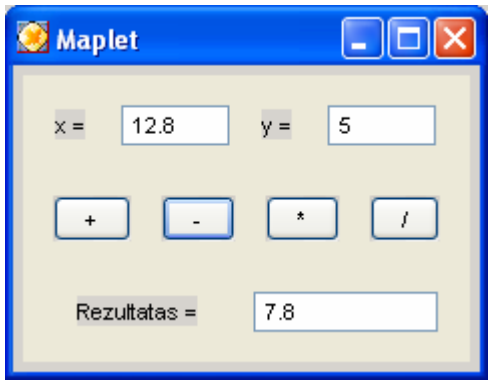
Komponentas	Įvykis	Vykdomas metodas
Button1 – Button4	Callback	pushbutton1_Callback – pushbutton4_Callback

Programų tekstai. Programų tekstai pateikti 2 priede.

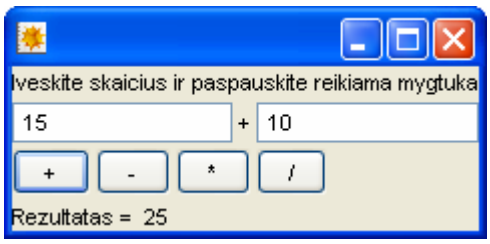
Programų darbo langai. Sistemose *Matlab* 7, *Maple* 10, *Mathematica* 5.2 ir *C++ Builder* 6 sukurtų programų, atliekančių skaičiuoklių funkciją, darbo langai parodyti 4.2 a, b, c paveiksluose.

a) 

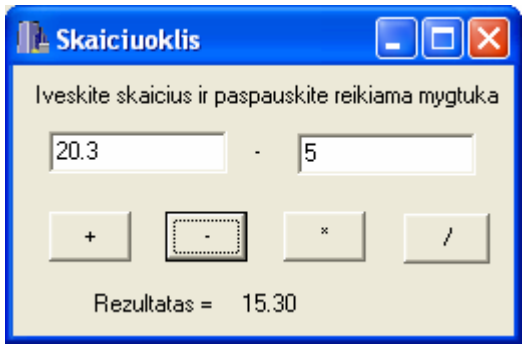
b)



c)



d)

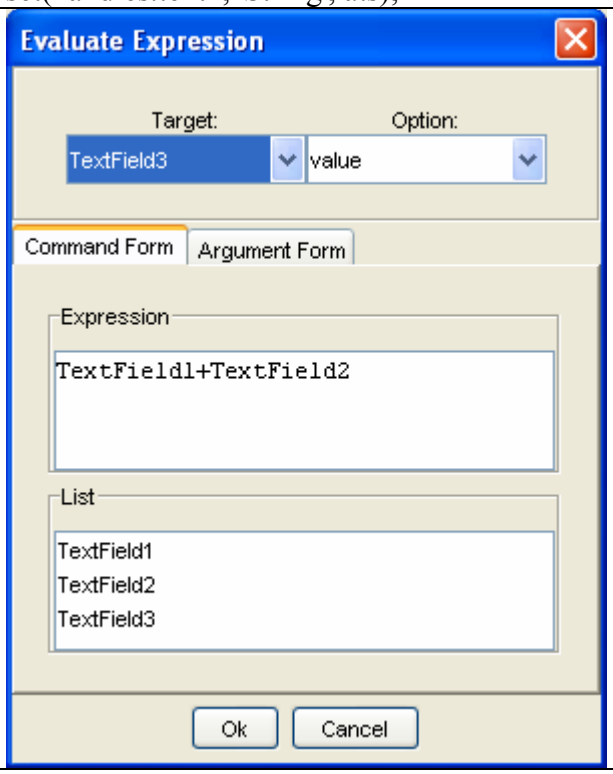


4.2 pav. Skaičiuoklis, sukurtas sistemomis *Matlab*, *Maple*, *Mathematica*, *C++ Builder*

Kodo kūrimo ypatumai. Vienas iš pagrindinių ir dažnai naudojamų programų komponentų yra redagavimo laukelis, skirtas duomenims iš vartotojo gauti, taip pat teksto laukelis, skirtas rezultatams išvesti. 4.4 lentelėje pateiktos komandos, paimančios duomenis iš redagavimo laukelių ir įrašančios rezultatus į tekstinius laukelius.

4.4 lentelė

Komandų užrašymas sistemos kalba

Programa	**Duomenų įvedimas**	**Rezultatų išvedimas**
Matlab 7	x = str2double(get(handles.edit1,'string'));	set(handles.text4, 'String', ats);
Maple 10	Įvedamoms reikšmėms nuskaityti specialaus skakinio nėra.	
Mathematica 5.2	x=ToExpression[PropertyValue[{"laukas1", "text"}]];	SetPropertyValue[{"etikete3","text"}, ToString[z, InputForm]];

Patebėjimai. Iš 4.4 lentelės matyti, kad *Matlab* ir *Mathematica* sistemose duomenų įvedimo ir rezultatų išvedimo būdai panašūs. Iš duomenų laukelio paimtos reikšmės formatą reikia transformuoti į realų skaičių (sistemoje *Matlab*) ar išraišką (sistemoje *Mathematica*) tam, kad būtų galima atlikti aritmetinius veiksmus su duomenimis. Išvedant rezultatus į tekstinį laukelį, būtina nurodyti komponento, į kurį išvedami rezultatai, savybę: sistemoje *Matlab* – String, sistemoje *Maple* – value, sistemoje *Mathematica* – text. *Matlab* sistemoje programos kodas rašomas **.m* byloje. *Mathematica* sistemoje kodas rašomas sistemos darbalaukyje. Sistemoje *Maple* rezultatų išvedimas į tekstinį laukelį labai skiriasi. *Maple* sistemoje formuojant reakciją į mygtuko paspaudimą, užpildoma *Evaluate Expression* kortelė – programos kodo rašyti nereikia.

***2 užduotis**. Tekstiniame duomenų faile duotas realiųjų skaičių dvimatis masyvas. Sukurti programą, kuri rastų didžiausią ir mažiausią masyvo elementus bei masyvo elementų vidurkį.*

Užduoties analizė. Šio uždavinio duomenys bus skaitomi iš tekstinio bylos dv_mas.txt. Pirmoje šios bylos eilutėje turi būti užrašytas eilučių skaičius n, antroje eilutėje – stulpelių skaičius m. Tolesnėse bylos eilutėse surašyti realūs skaičiai, atskirti tarpais. Duomenų bylos pavyzdys:

```
5
4
14.2   5.2   8.5   4.2
 7.3  12.0   4.3  16.7
 4.0   1.1  14.4  25.0
45.1   2.2   8.4   4.9
 5.1  23.0   7.3  15.1
```

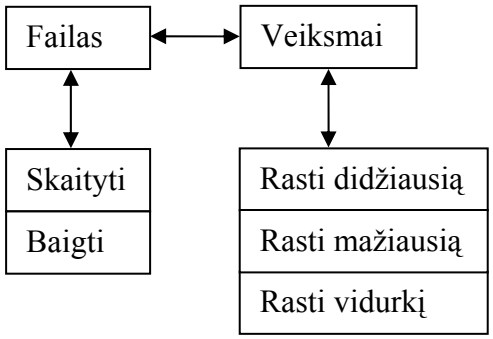
4.3 pav. Duomenų bylos sistemoje Maple pavyzdys.

Programa turės atlikti 5 skirtingus metodus:

a) nuskaityti duomenis iš bylos,
b) rasti dvimačio masyvo didžiausią elementą,
c) rasti dvimačio masyvo mažiausią elementą,
d) rasti masyvo elementų vidurkį,
e) baigti darbą.

Sudarant programas, kuriose atliekamų veiksmų skaičius didesnis, joms valdyti geriausiai parengti meniu. Bus naudojamas horizontalusis meniu, esantis programos lango viršuje.

Programos meniu schema pateikta 4.4 paveiksle.



4.4 pav. Programos „Dvimatis masyvas“ meniu schema.

4.5 lentelė

Komponentų paskirtis

Vardas	Paskirtis
StaticText1	Rezultatų langelis
StaticText2	Paaiškinimui išvesti
StaticText3	Masyvo elementams išvesti

4.6 lentelė

Savybių lange keičiamos komponentų savybių reikšmės

Vardas	Keičiama savybė	Reikšmė
StaticText1	String	
StaticText2	String	Pasirinkite veiksmus iš meniu.
StaticText3	String	

4.7 lentelė

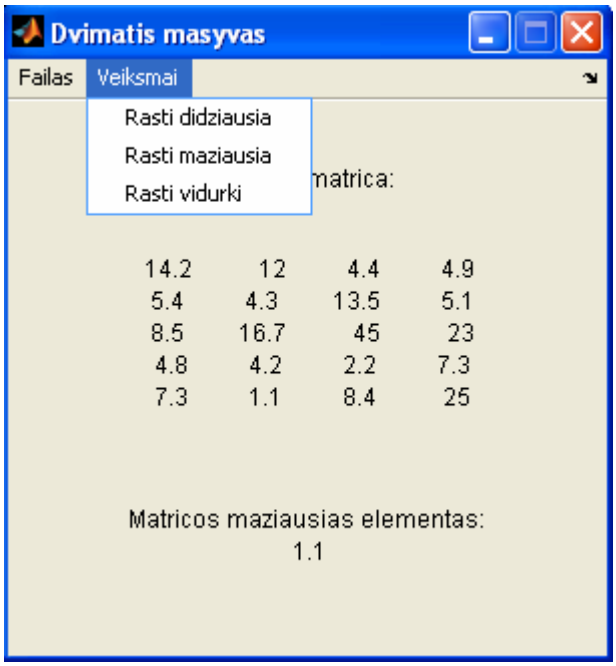
Įvykiai, į kuriuos reaguoja programos komponentai

Meniu punktas	Įvykis	Vykdomas metodas
Skaityti	Callback	menu_failas_skaityti_Callback
Baigti	Callback	menu_failas_baigti_Callback
Rasti didžiausią	Callback	menu_veiksmai_didziausias_Callback
Rasti mažiausią	Callback	menu_veiksmai_maziausias_Callback
Rasti vidurkį	Callback	menu_veiksmai_vidurkis_Callback

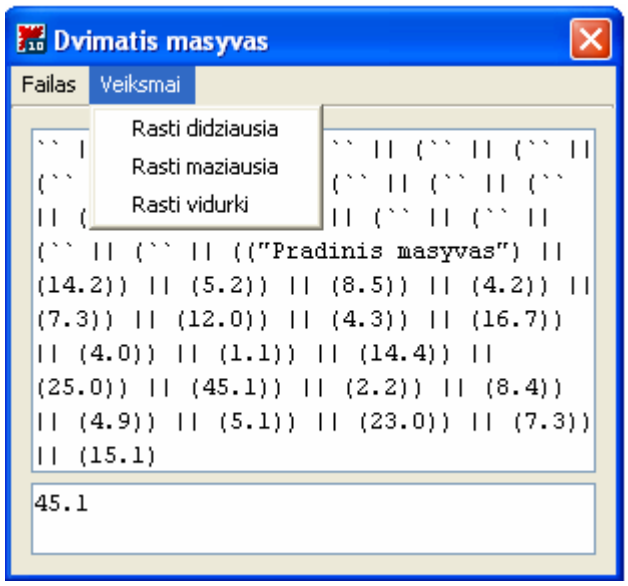
Programų tekstai pateikti 2 priede.

Programų darbo langai.

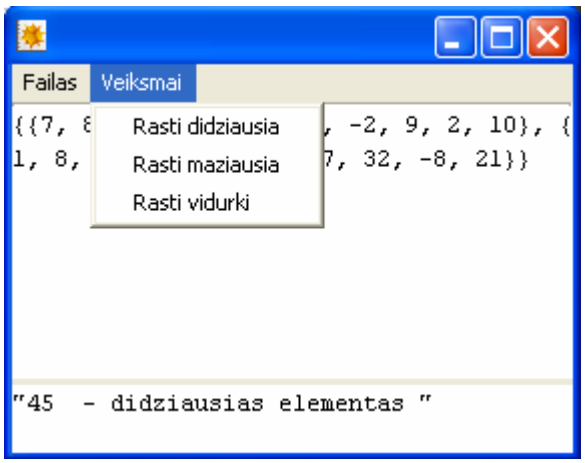
a)

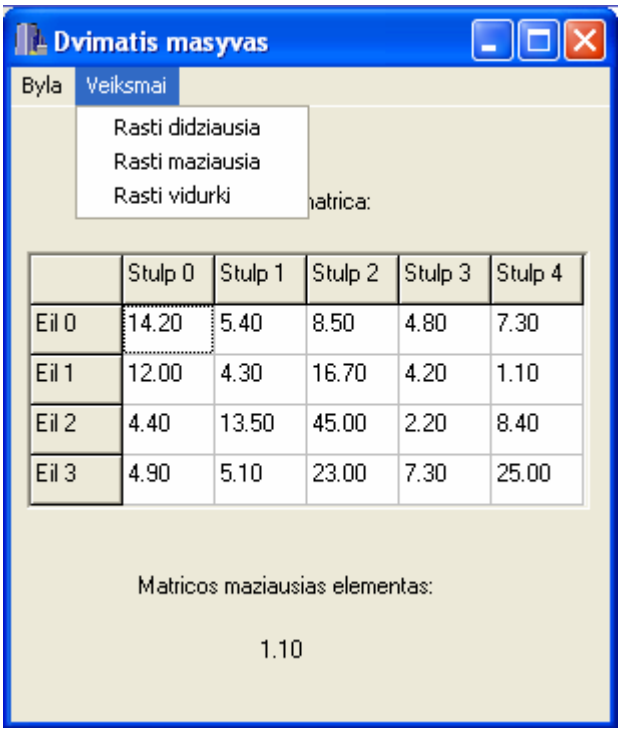

Dvimatis masyvas
Failas
Veiksmai
Rasti didziausia
Rasti maziausia
Rasti vidurki
matrica:
14.2 12 4.4 4.9
5.4 4.3 13.5 5.1
8.5 16.7 45 23
4.8 4.2 2.2 7.3
7.3 1.1 8.4 25
Matricos maziausias elementas:
1.1

b)


Dvimatis masyvas
Failas
Veiksmai
Rasti didziausia
Rasti maziausia
Rasti vidurki
(("Pradinis masyvas") ||
(14.2)) || (5.2)) || (8.5)) || (4.2)) ||
(7.3)) || (12.0)) || (4.3)) || (16.7))
|| (4.0)) || (1.1)) || (14.4)) ||
(25.0)) || (45.1)) || (2.2)) || (8.4))
|| (4.9)) || (5.1)) || (23.0)) || (7.3))
|| (15.1)
45.1

c)


Failas
Veiksmai
Rasti didziausia
Rasti maziausia
Rasti vidurki
, -2, 9, 2, 10}, {
7, 32, -8, 21}}
"45 - didziausias elementas "

d) 

4.5 pav. Programų „Dvimatis masyvas“ darbo langai, sukurti sistemose a) Matlab, b) Maple, c) Mathematica ir d) C++ Builder

Kodo kūrimo ypatumai. Sistemoje *Matlab* 7 yra patogus ir greitas būdas masyvo didžiausiam, mažiausiam elementams rasti bei vidurkiams apskaičiuoti – standartinių funkcijų naudojimas (min, max, average). Komanda užrašoma *did = max(max(A))* – didžiausias elementas randamas kiekviename stulpelyje, o tada iš gautų reikšmių vėl randama didžiausia. Sistemose *Maple* 10 ir *Mathematica* 5.2 rezultatus galima gauti tik klasikiniu būdu – užrašant algoritmą su žinomo kartojimų skaičiaus ciklu – panašiai kaip ir sistemoje *C++ Builder* 6.

Sistema *Mathematica* apskaičiuotą vidurkį išveda trumpenos pavidalu. Todėl naudojama funkcija *N[vid]*.

Pastebėjimai. Iš pateiktų programų galima padaryti išvadą, kad klasikinius programavimo uždavinius su masyvais galima realizuoti KMS. Meniu išvaizdos grafinė vartotojo sąsaja gali būti realizuota taip pat kiekvienoje KMS. Sistemų meniu kūrimo ypatumai pateikti 4.8 lentelėje. Žinoma, sąsajos išvaizda kiekvienoje sistemoje šiek tiek skiriasi. Sistemoje *Matlab* 7 galima sukurti grafinę vartotojo sąsają, panašiausią į sistemos *C++ Bulider* 6 sistemoje sukurtą. Sistemose *Maple* 10 ir *Mathematica* 5.2 sukurtos programos sąsajos vaizdas gan panašus. Skirtumus tarp sistemų programų grafinės vartotojo sąsajos lemia komponentų palečių skirtumai (*C++ Builder* sistemoje komponentų

paletės kur kas turtingesnės nei KMS). Visose nagrinėjamose KMS nėra *StringGrid* komponento, naudojamo dvimačiam masyvui išvesti.

4.8 lentelė

Meniu kūrimo ypatumai KMS

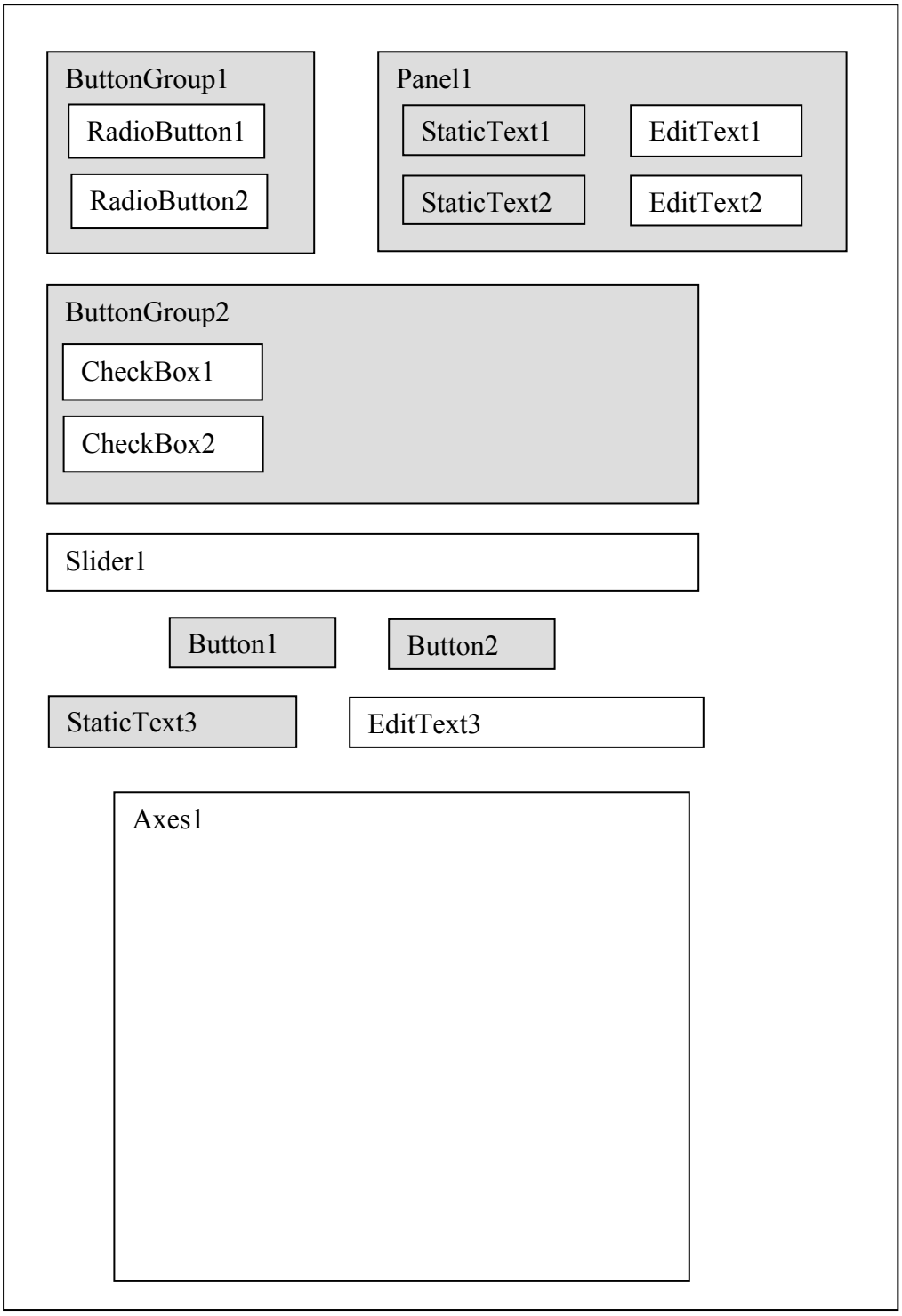
Sistema	**Matlab 7**	**Maple 10**	**Mathematica 5.2**
Meniu kūrimo ypatumai	Meniu sukurtas, naudojantis įrankiu Tools->Menu Editor	Meniu sukurtas, meniu elementus aprašant sistemos Maple 10 programavimo kalbos kodu.	Meniu sukurtas, meniu elementus aprašant sistemos Mathematica 5.2 programavimo kalbos kodu.

***3 uždavinys**. Duota funkcija f(x). Parašyti programą, kuri apskaičiuotų duotos funkcijos išvestinę ir neapibrėžtinį integralą, nubrėžtų duotos funkcijos ir rezultatų grafikus.*

Užduoties analizė. Grafinės vartotojo sąsajos kūrimo tyrimams pratęsti pasirenkamas pavyzdys, reikalaujantis analizinių skaičiavimų, grafikų braižymo, veiksmo indikatoriaus komponento panaudojimo. Tokį uždavinį suprogramuoti naudojantis kuria nors universalia programavimo kalba būtų žymiai sunkiau, o imant sunkesnius funkcijų atvejus praktiškai neįmanoma.

Funkcijai ir kintamajam, pagal kurį reikės diferencijuoti/integruoti, įvesti bus panaudoti teksto eilutės (redagavimo) komponentai. Programa kuriama taip, kad vartotojas pats galėtų pasirinkti vieną iš dviejų galimų veiksmų: diferencijuoti ar integruoti. Toliau vartotojas gali pasirinkti: brėžti įvestos funkcijos grafiką, diferencijuotos/integruotos funkcijos grafiką ar abu grafikus iš karto. Šiam pasirinkimui realizuoti labai gerai tinka veiksmo indikatoriaus komponentas (*CheckBox*). Grafiko abscisių ašies viršutinę ribą gali koreguoti slankiklio komponentas (*Slider*).

Programos formos schema



4.6 pav. Programos „Diferencijavimas-Integravimas“ formos schema.

4.9 lentelė

Komponentų paskirtis

Vardas	Paskirtis
ButtonGroup1, ButtonGroup2, Panel1	Panašiems užrašams aprėminti.
RadioButton1	Diferencijavimo operacijai pažymėti.
RadioButton2	Integravimo operacijai pažymėti.
StaticText1, StaticText2, StaticText3	Programos paaiškinimų langeliai.
EditText1	Funkcijai įvesti.
EditText2	Kintamajam, pagal kurį bus diferencijuojama/integruojama įvesti.
CheckBox1	Funkcijos grafikui pažymėti.
CheckBox2	Diferencijuotos/integruotos funkcijos grafikui pažymėti.
EditText3	Rezultatui išvesti
Axes1	Grafikui parodyti.

4.10 lentelė

Savybių lange keičiamos komponentų savybių reikšmės

Vardas	Keičiama savybė	Reikšmė
ButtonGroup1 – 2	Title	Pasirinkite
RadioButton1	String	Diferencijuoti
RadioButton2	String	integruoti
Panel1	Title	Iveskite
StaticText1	String	Iveskite funkciją
StaticText2	String	diferencijuoti pagal kintamąjį
StaticText3	String	Rezultatas:
EditText1	String	x^5
EditText2	String	x
Button1	String	Veiksmai
Button2	String	Baigti
EditText3	String	

4.11 lentelė

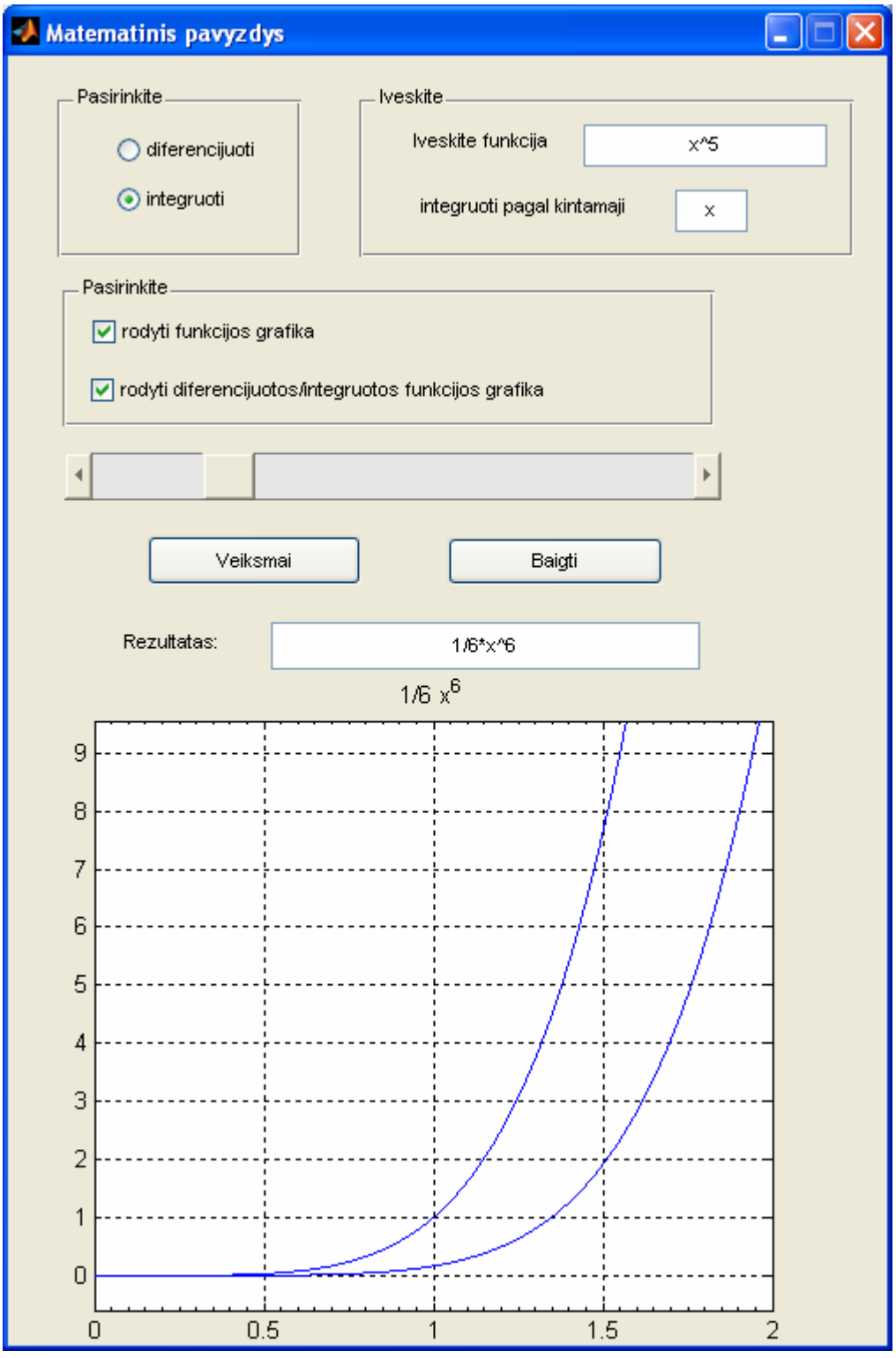
Įvykiai, į kuriuos reaguoja programos komponentai

Komponentas	Įvykis	Vykdomas metodas
RadioButton1 – RadioButton2	Callback	radiobutton1_Callback – radiobutton2_Callback
CheckBox1 – CheckBox2	Callback	checkbox1_Callback – checkbox2_Callback
Slider1	Callback	slider1_Callback
Button1 – Button2	Callback	pushbutton1_Callback – pushbutton2_Callback

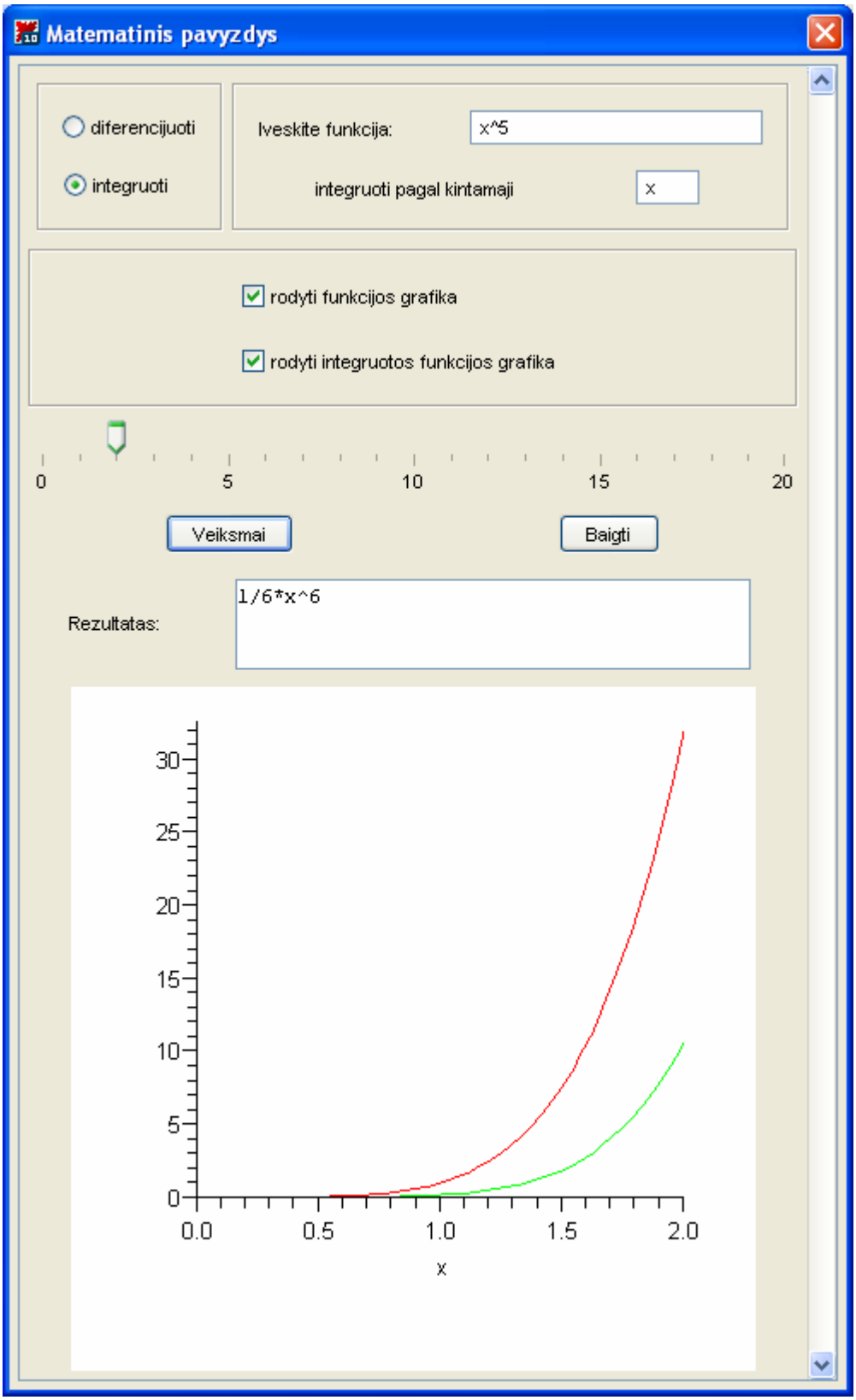
Programų tekstai pateikti 2 priede.

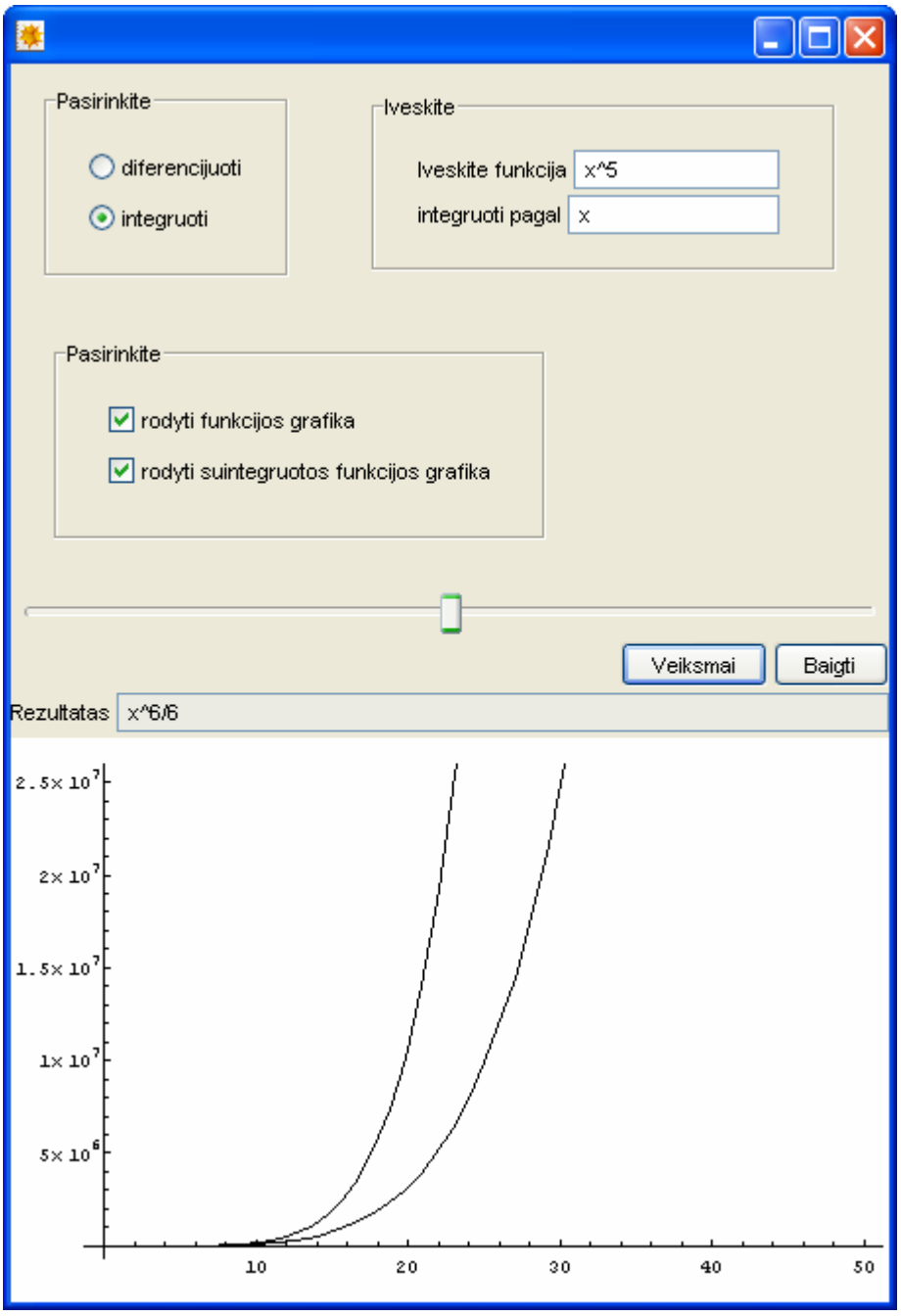
Programų darbo langai.

a)


Matematinis pavyzdys
Pasirinkite
diferencijuoti
integruoti
Iveskite
Iveskite funkcija
x^5
integruoti pagal kintamaji
x
Pasirinkite
rodyti funkcijos grafika
rodyti diferencijuotos/integruotos funkcijos grafika
Veiksmai
Baigti
Rezultatas:
1/6*x^6
1/6 x^6
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2

b)


Matematinis pavyzdys
diferencijuoti
integruoti
Iveskite funkcija:
x^5
integruoti pagal kintamaji
x
rodyti funkcijos grafika
rodyti integruotos funkcijos grafika
0
5
10
15
20
Veiksmai
Baigti
Rezultatas:
1/6*x^6
30
25
20
15
10
5
0
0.0
0.5
1.0
1.5
2.0
x

c) 

4.7 pav. Uždavinys „Diferencijavimas-Integravimas“, sukurtas sistemose a) Matlab 7, b) Maple 10, c) Mathematica 5.2

Kodo kūrimo ypatumai. Sistemoje *Matlab* 7, priešingai nei kitose nagrinėjamose sistemose, prieš diferencijuojant/integruojant funkciją, reikia aprašyti kintamąjį, pagal kurį diferencijuojama/integruojama, kaip simbolinį. Tai atliekama komanda *k = sym(kint)*. *Matlab* 7 sistemoje šiam uždaviniui spręsti labai tiko sąlygos sakinys su *elseif* ir *else* šakomis pasirinkimams iš veiksmo indikatoriaus komponentų nustatyti.

Sistemoje *Maple* 10 specifiškai paimama reikšmė iš veiksmo indikatoriaus komponento: *pas1:=Maplets:-Tools:-Get('ChB1');*

Pastebėjimai. Iš 4.7, 4.8, 4.9 paveikslėlių matyti, kad panašią grafinę vartotojo sąsają galima sukurti su visomis kompiuterinėmis matematikos sistemomis. Programa sistemoje *Matlab* 7 buvo sukurta vaizdžiuoju būdu sudedant komponentus ir programuojant tik reakciją į įvykius. Sistemose *Maple* 10 ir *Mathematica* 5.2 sąsajos sukurtos aprašant programavimo kalbomis ir pačius komponentus, ir reakciją į įvykius.

4 uždavinys. *Tekstiniame duomenų faile duoti tokie duomenys: vardas, pavardė, pažymiai. Apskaičiuojamas kiekvieno studento vidurkis. Įrašų masyvas surikiuojamas ir rezultatai išvedami į tekstinį failą.*

Užduoties analizė. Šio uždavinio duomenys bus skaitomi iš tekstinės bylos. Duomenys byloje surašyti tokia tvarka: vardas (9 pozicijos), pavardė (15 pozicijų), pažymiai (natūralūs skaičiai), atskirti bent vienu tarpo simboliu. Duomenų bylos pavyzdys:

Petras	Balcytis	9 10 8 9
Jonas	Mileris	5 7 6 4
Maryte	Dauguvietyte	8 7 9 7
Ugne	Silkaityte	8 10 9 8

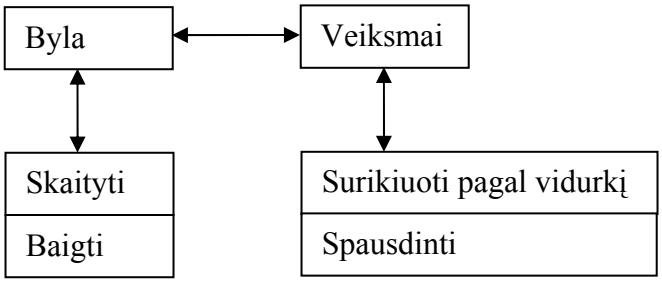
4.8 pav. Duomenų bylos sistemoje Matlab pavyzdys.

Programa turės atlikti 5 skirtingus metodus:

a) nuskaityti duomenis iš bylos,
b) rasti pažymių vidurkį,
c) surikiuoti įrašų masyvą mažėjimo tvarka,
d) atspausdinti rezultatus į tekstinę bylą,
e) baigti darbą.

Bus naudojamas horizontalusis meniu, esantis programos lango viršuje.

Programos meniu schema pateikta 4.9 paveiksle.



4.9 pav. Programos „Studentai“ meniu schema.

4.12 lentelė

Komponentų paskirtis

Vardas	Paskirtis
StaticText1	Paaiškinimų langelis
StaticText2	Įrašų masyvui išvesti

4.13 lentelė

Savybių lange keičiamos komponentų savybių reikšmės

Vardas	Keičiama savybė	Reikšmė
StaticText1	String	Pasirinkite veiksmus iš meniu.
StaticText2	String	

4.14 lentelė

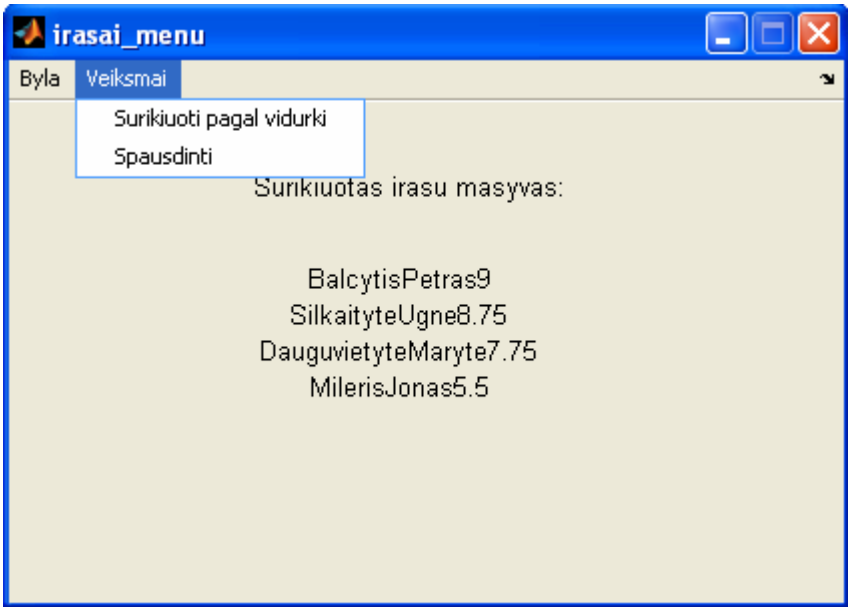
Įvykiai, į kuriuos reaguoja programos komponentai

Meniu punktas	Įvykis	Vykdomas metodas
Skaityti	Callback	menu_failas_skaityti_Callback
Baigti	Callback	menu_failas_baigti_Callback
Rikiuoti pagal vidurkį	Callback	menu_veiksmai_surikiuoti_Callback
Spausdinti	Callback	menu_veiksmai_spausdinti_Callback

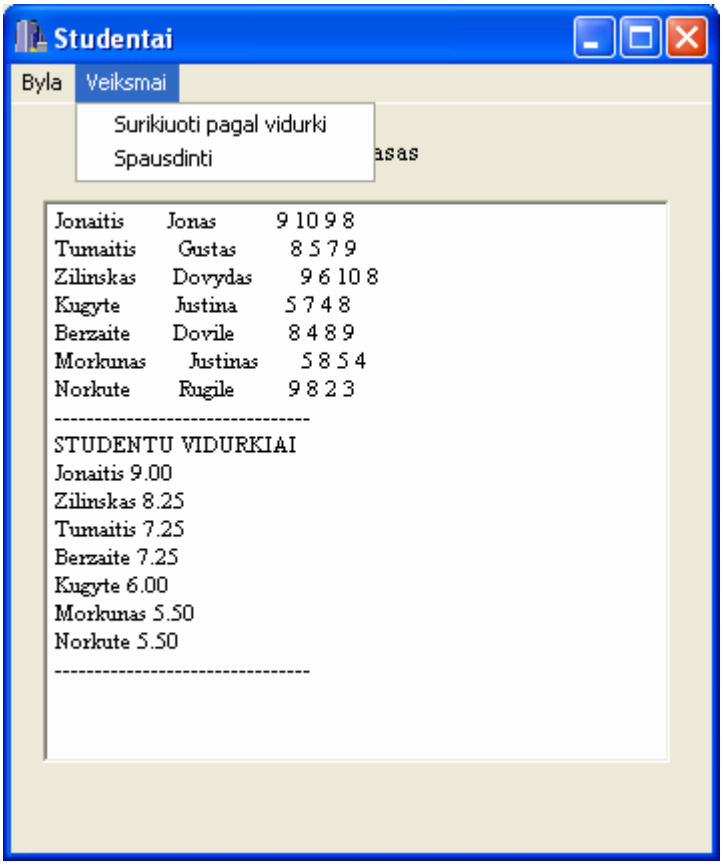
Programų tekstai pateikti 2 priede.

Programų darbo langai.

a)



b)



4.10 pav. Programos „Studentai“ darbo langai, sukurti sistemose a) Matlab 7 ir b) C++ Builder 6

Kodo kūrimo ypatumai. Sistemoje *Matlab* 7 programos, atliekančios įrašų masyvo nuskaitymą iš bylos turi panašumų su analogiškomis sistemos *C++ Builder* 6 programomis. Įdomesnis nuskaitymo sakinys, kuriame reikia suskaičiuoti, kiek bus to paties tipo elementų: *[A(n).vard,count] = fscanf(FD,'%9s',1);*

Įrašų masyvo išvedimui į komponentą, naudojama funkcija *strcat*, sujungianti įrašo laukus į vieną eilutę: *strcat(A(i).pav,' ', A(i).vard,' ', num2str(A(i).vid))*.

Pastebėjimai. Sistemose *Mathematica* 5.2 ir *Maple* 10 yra galimybė nuskaityti duomenis iš bylos į įrašų masyvus, tačiau kol kas nėra galimybės atlikti veiksmus su įrašo laukais.

4.3. KMS PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBIŲ PALYGINIMAS

Iš pateiktų programų grafinės vartotojos sąsajos pavyzdžių, galima suformuluoti tokius KMS kūrimo galimybių skirtumus:

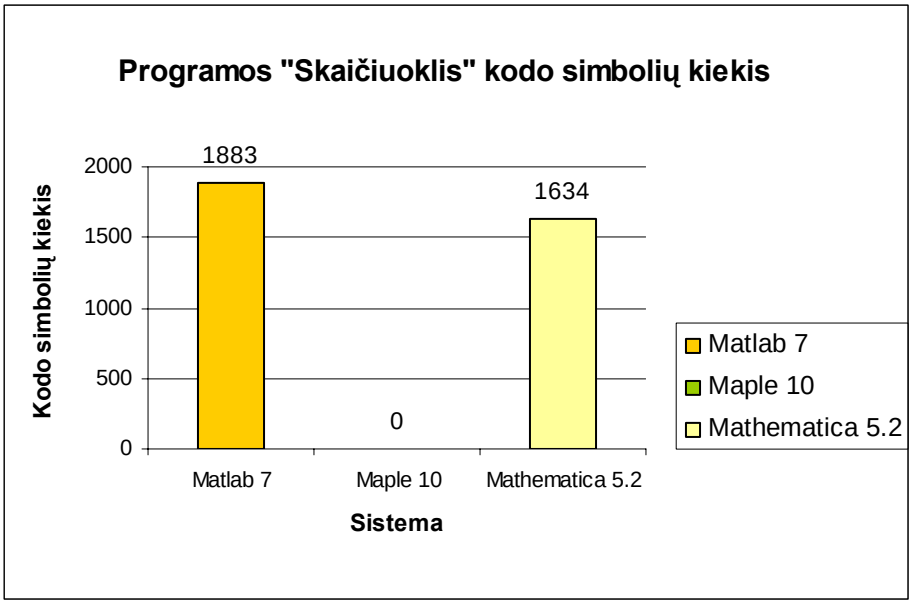
1. Skiriasi grafinės vartotojo sąsajos kūrimo procesas. Sistemose *Matlab* 7 ir *Maple* 10 programa konstruojama iš vaizdinių komponentų, kurie išrenkami iš komponentų palečių ir pele įkeliami į sąsajos šabloną. Sistemoje *Mathematica* 5.2 tokios galimybės dar nėra. Čia grafinė vartoto sąsaja kuriama sistemos vidine programavimo kalba aprašant komponentus. Aukštesnio lygio elementas gali turėti vieną ar kelis žemesnio lygio elementus [2].
2. Lyginant sąsajos kūrimo procesą vaizdžiuoju būdu sistemose *Matlab* 7 ir *Maple* 10, pastarojoje sistemoje kol kas neįmanoma sukurti sudėtingesnių programų, nes nėra galimybės naudoti procedūras. Todėl dauguma šio darbo programų sukurta aprašant elementus sistemos vidine programavimo kalba.
3. Skirtingas sukuriamų bylų skaičius ir programų vykdymo galimybės. Šios ypatybės pateiktos 4.15 lentelėje.

4.15 lentelė

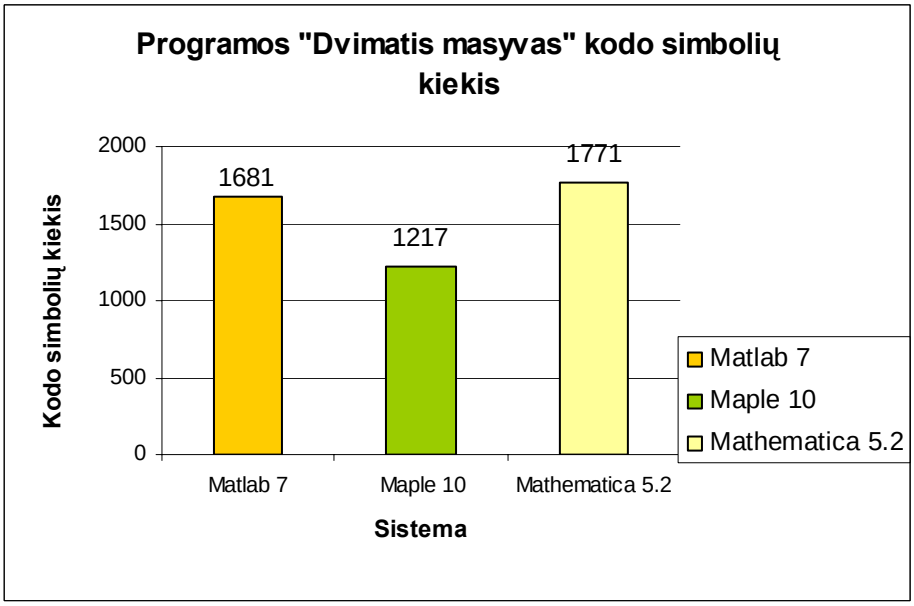
Programų saugojimas ir vykdymas

	Matlab 7	**Maple 10**	**Mathematica 5.2**
Kūrimas rašant kodą	Programos tekstas rašomas sistemos *Matlab* komandų lange arba m-faile.	Programos tekstas rašomas sistemos *Maple* darbalaukyje arba komandų eilutėje. Sukurta programa išsaugoma byloje su prievardžiu *mw*.	Programos kodas rašomas sistemos *Mathematica* darbalaukyje ir išsaugomas byloje su prievardžiu *nb*.
„Vaizdusis“ būdas	Programos tekstas rašomas byloje su prievardžiu *m*. Grafinis programos vaizdas išsaugomas byloje su prievardžiu *fig*. Sukurta programa paleidžiama iš grafinės vartotojo sąsajos kūrimo terpės GUIDE.	Programos grafinis vaizdas, komponentų savybės ir atliekami veiksmai išsaugomi byloje su prievardžiu *maplet*. Sukurtą programą galima tiesiogiai vykdyti, nepaleidus *Maple* sistemos.	-

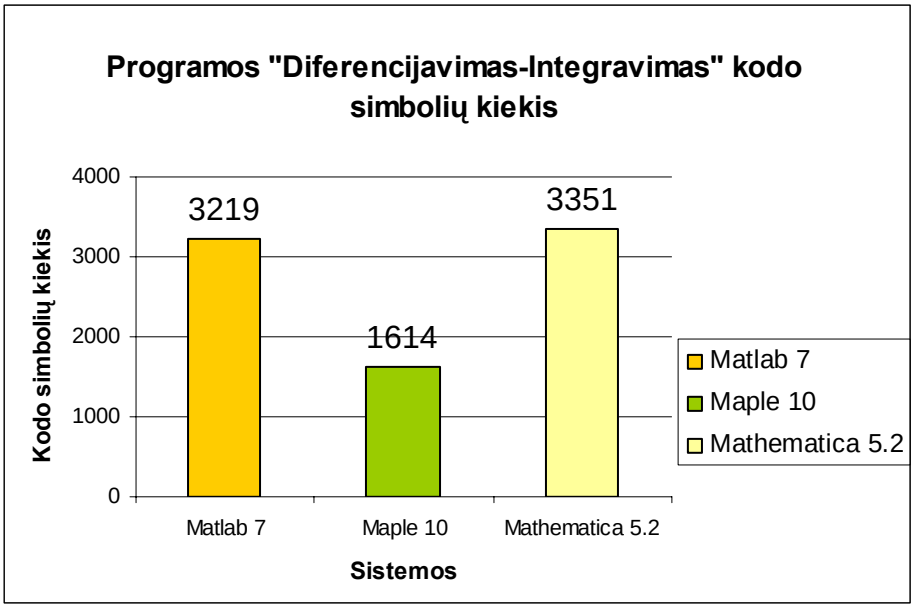
4. Skiriasi programos kodo apimtis. Programos „Skaičiuoklis“ didžiausias simbolių kiekis yra sistemos *Matlab* 7 *m* faile, tačiau kai kurios kodo eilutės (pvz., metodų aprašai) yra sugeneruotos pačios sistemos – programuojama tik reakcija į įvykius. Sistemoje *Maple* 10 programa sukurta naudojant *Maplet Builder* paketą, todėl programuotojui kodo rašyti nereikia . Reakcija į įvykius aprašoma užpildžius savybių nustatymo korteles (pvz., *Evaluate Expression* kortelę kuri pateikta 4.1. lentelėje). Programose „Dvimatis masyvas“ bei „Diferencijavimas-Integravimas“ didžiausias simbolių kiekis yra sistemos *Mathematica* 5.2 programose. Lyginant visų trijų programų simbolių kiekį, galima padaryti išvadą, kad sistemoje *Maple* 10 programos kodo apimtis yra gerokai mažesnė lyginant su kitomis KMS. Sukurtų programų simbolių kiekių palyginimas pateiktas 4.11, 4.12, 4.13 paveiksluose (čia lyginamas sistemos *Matlab* 7 bylos su prievardžiu *m* , sistemos *Maple* 10 bylos su prievardžiu *mw*, sistemos *Mathematica* 5.2 bylos su prievardžiu *nb* simbolių skaičius).



4.11 pav. Programos „Skaičiuoklis“ kodo simbolių kiekis



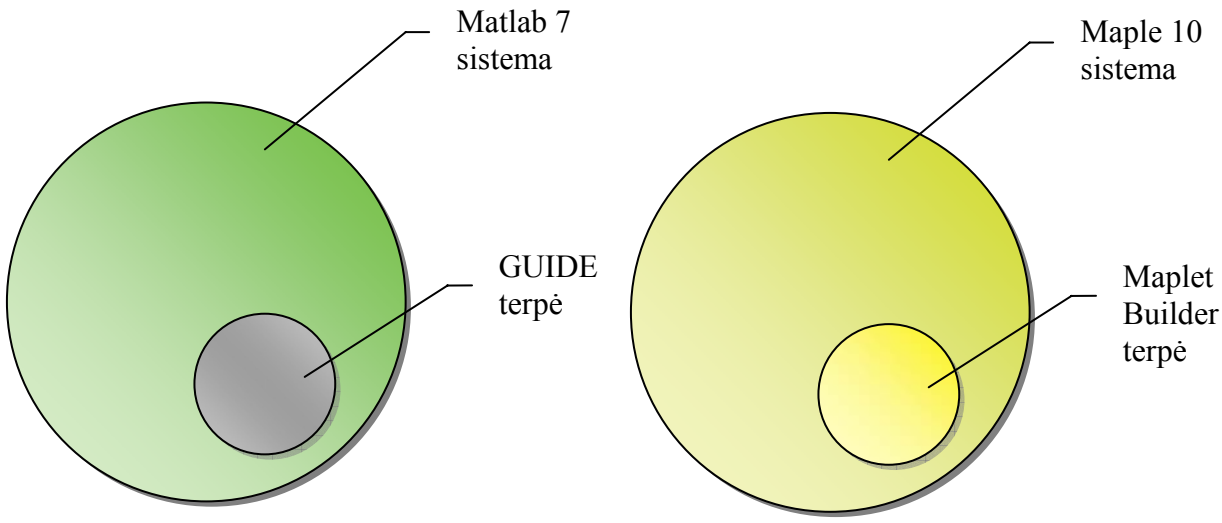
4.12 pav. Programos „Dvimatis masyvas“ kodo simbolių kiekis



4.13 pav. Programoje „Diferencijavimas-Integravimas“ kodo simbolių kiekis

KMS grafinės vartotojo sąsajos kūrimo galimybių panašumai:

1. Yra galimybė kurti grafinę vartotojo sąsają kiekvienoje KMS [2].
2. Programos grafinės vartotojo sąsajos kūrimui vaizdžiuoju būdu skirtos terpės, panašios savo įrankiais, priemonėmis ir galimybėmis. Šių terpių sąsaja su sistemomis *Matlab* 7 ir *Maple* 10 pavaizduota 4.14 paveiksle.



4.14 pav. Grafinės vartotojo sąsajos kūrimo terpės ryšys su sistema.

3. Sąsajai kurti naudojami panašūs komponentai. Netgi kai kurių elementų pavadinimai kiekvienoje KMS sutampa, pvz., ListBox, RadioButton, CheckBox [2]. Išsamus komponentų aprašymas pateiktas lentelėje 1 priede.
4. Kiekvienas elementas charakterizuojamas savybėmis, metodais, įvykiais [2].
5. Sistemose *Mathematica 5.2* ir *Maple 10* panašus komponentus aprašančio kodo kūrimo principas: aukštesnio lygio komponentas gali turėti kelis žemesnio lygio komponentus – atvirkščias teiginys nėra teisingas [2].
6. Kiekvienoje KMS galima sukurti panašios išvaizdos grafinę vartotojo sąsają (4.8, 4.9, 4.10 pav.) [2].

4.4. KMS IR C++ BUILDER PROGRAMŲ GRAFINĖS VARTOTOJO SĄSAJOS KŪRIMO GALIMYBIŲ PALYGINIMAS

Kiekvienoje sistemoje grafinės vartotojo sąsajos kūrimas turi savų ypatybių ir niuansų. Nors *C++ Builder* sistema savo paskirtimi labai skiriasi nuo KMS, tačiau šias sistemas vienija bendras bruožas – galimybė kurti patogią ir patrauklią programos grafinę vartotojo sąsają.

Programų grafinės vartotojo sąsajos kūrimo galimybių nagrinėjimas KMS – daug laiko sąnaudų reikalaujantis darbas. Tačiau mokančiam kurti sąsają *C++ Builder* sistema, mokytis kurti grafinę vartotojo sąsają KMS lyginimo metodu yra žymiai lengviau [2].

KMS ir *C++ Builder* grafinės vartotojo sąsajos kūrimo galimybių skirtumai:

1. Skiriasi grafinės vartotojo sąsajos kūrimo procesu [2]. Sistemose *C++ Builder* 6, *Matlab* 7, *Maple* 10 sąsaja konstruojama vaizdžiuoju būdu – pele tempiant komponentus į programos langą šabloną ir programuojant tik reakciją į įvykius (komponentų savybės nustatomos specialiose kortelėse). Sistemoje *Mathematica* 5.2 kol kas grafinė vartotojo sąsaja konstruojama aprašant ir komponentus, ir metodus.
2. Sistemose *C++ Buider* 6 ir *Maple* 10 skiriasi programų kūrimo būdas. *C++ Builder* 6 sistemoje grafinis vaizdas išsaugomas byloje su prievardžiu *.dfm* [12], o pagrindinis programos tekstas saugomas byloje su prievardžiu *.cpp*. Sistemoje *Maple* 10 atskiro failo programos kodui nėra. Reakcija į įvykius aprašoma užpildžius tam tikras korteles (pvz., *Evaluate Expression* kortelę – ši kortelė iškviečiama pildant komponentų savybių kortelę). Todėl kol kas aprašyti keletą veiksmų (tarkim mygtuko paspaudimą ir savybės pakeitimą)

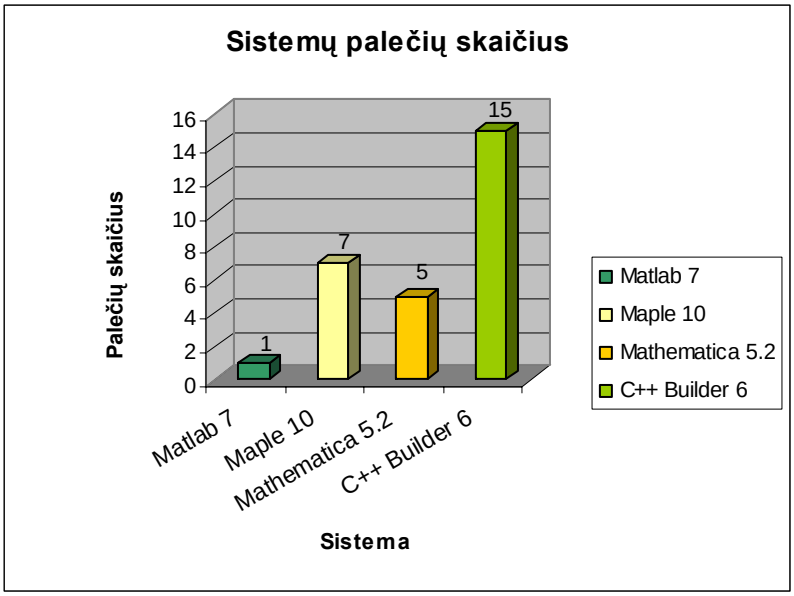
nėra galimybės. Tačiau sudėtingesnes programas galima sukurti senesniuoju būdu aprašant komponentus ir įvykius byloje su prievardžiu *.mw*.

3. Sistema *C++ Builder* 6 sukurtą programą galima paleisti įvykdžius bylą su prievardžiu *.exe* – nebūtina paleisti ar turėti kompiuteryje *C++ Builder* 6 sistemą. Sistema *Maple* 10 sukurta programa gali būti paleista įvykdžius bylą su prievardžiu *.maplet* – pati sistema *Maple* 10 nėra reikalinga. Sistema *Mathematica* 5.2 sukurtą programą galima paleisti tik įjungus sistemą *Mahematica* 5.2 ir tik iš jos pagrindinio darbalaukio. Sistemos *Matlab 7* programai taip pat reikia *Matlab 7* terpės.
4. Vartoja skirtingas programavimo kalbas. *C++ Builder* sistema vartoja universalią objektinio programavimo kalbą *C++*, o KMS – specializuotas vidines procedūrinio programavimo kalbas, kurių sintaksė artima *Pascal* ir *C* kalboms[2].
5. Tos pačios programos grafinės vartoto sąsajos vaizdo skirtumus lemia nevienodas palečių kiekis ir juose esančių komponentų skaičius (4.16 lentelė ir 1 priede esanti lentelė). Palečių ir komponentų skaičiaus skirtumai pavaizduoti 4.15 ir 4.16 paveiksluose.

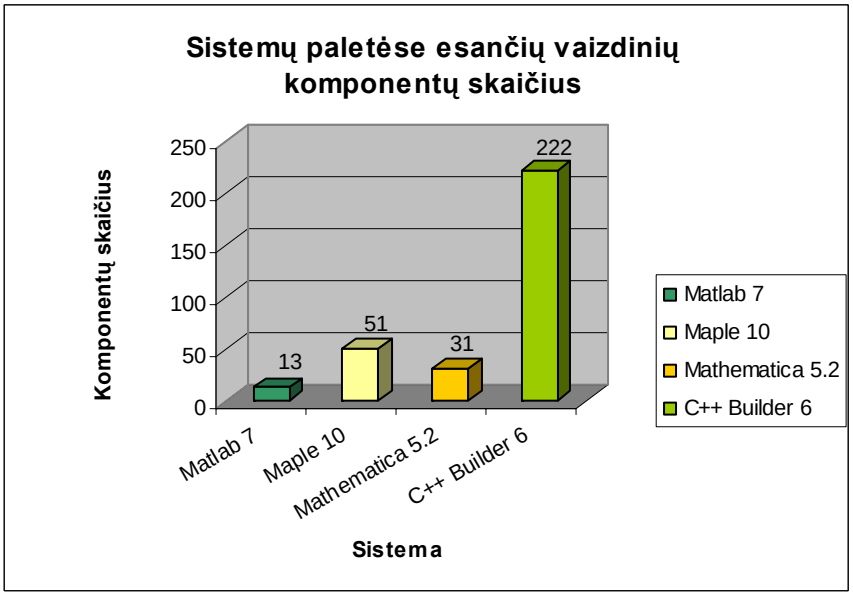
4.16 lentelė

Palečių skirtumai

Sistema	**Paletės**
Matlab 7	Viena pagrindinė komponentų paletė – 13 komponentų.
Maple 10	Paletės: Body (15 komponentų), Dialog (7), Menu (7), Toolbar (3), Other (7), Layout (6), Command (6). Iš viso 51 komponentas.
Mathematica 5.2	Paletės nėra.
C++ Builder 6	Paletės: Standard (16 komponentų), Additional (15), Win32 (18), System (8), Data Access (8), Data Controls (15), InterBase (11), Internet (8), FastNet (18), QReport (23), Dialogs (10), Win 3.1 (11), Samples (10), ActiveX (4), Servers (47). Iš viso 222 komponentai.



4.15 pav. Sistemų palečių skaičius



4.16 pav. Sistemų paletėse esančių vaizdinių komponentų skaičius

5. IŠVADOS

1. Tos pačios programos grafinis vaizdas, sukurtas nagrinėjamose KMS, gali būti labai panašus, tačiau gali skirtis dėl sistemų palečių bei juose esančių komponentų neatitikimo.
2. Visomis nagrinėtomis KMS galima realizuoti šiuos uždavinius: skaitymą iš tekstinės bylos, įvairius veiksmus su vienmačio ir dvimačio masyvo elementais, skaitymą iš bylos į įrašų masyvą, tačiau atlikti veiksmus su įrašų masyvo laukais kol kas įmanoma tik *Matlab* 7 sistemoje – sistemose *Maple* 10 ir *Mathematica* 5.2 tokios galimybės kol kas nėra.
3. Šiuo metu grafinę vartotojo sąsają lengviausiai kurti *Matlab* 7 sistemoje lyginant su kitomis nagrinėjamomis sistemomis, nes
    a) yra galimybė grafinę vartotojo sasają kurti vaizdžiuoju būdu, todėl nereikia aprašinėti komponentų kaip sistemoje *Mathematica* 5.2;
    b) programų kūrimo būdas sistemose *Matlab* 7 ir *C++ Builder* 6 panašiausias lyginant su kitomis sistemomis;
    c) kuriant vaizdžiuoju būdu *Matlab* 7 sistemoje galima sukurti sudėtingesnių programų nei *Maple* 10 sistemoje (pastarojoje sistemoje kol kas nėra galimybės panaudoti procedūrų).
4. KMS tiktų ne informatikos specialybės studentams programavimui mokytis, nes jose yra grafinės sąsajos kūrimo galimybės, visos programavimo konstrukcijos bei matematinis aparatas.

6. LITERATŪRA

1. Turskienė S. Kompiuterinių matematikos sistemų programavimo galimybės. Šiauliai: Šiaulių universiteto leidykla, 2004.- 134 p.
2. Turskienė S. Kompiuterinių matematikos sistemų programų grafinės vartotojo sąsajos kūrimo galimybių lyginamoji analizė. Informacijos mokslai, t. 34, 2005.- 335-339 p.
3. Turskienė S. Kompiuterinių matematikos sistemų programavimo kalbų lyginamoji analizė. Lietuvos matematikos rinkinys, t. 44, spec. nr., 2004.- 377-382 p.
4. http://www.mathworks.com/access/helpdesk/help/techdoc/creating_guis/creating_guis.html [žiūrėta 2004-11-04]
5. http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_prog/bqjgwp9.html [žiūrėta 2005-02-10]
6. http://www.maplesoft.com/demo/recorded-seminars.aspx [žiūrėta 2005-11-04]
7. http://www.maplesoft.com/products/maple/touchmath.aspx# [žiūrėta 2006-01-15]
8. http://documents.wolfram.com/mathematica/Add-onsLinks/GUIKit/ [žiūrėta 2005-11-23]
9. http://documents.wolfram.com/pdf/AddendumV51.pdf [žiūrėta 2006-02-18]
10. Blonskis J., Vidžiūnas A.. ir kt. Delphi 5 programavimo pavyzdžiai. Kaunas: Smaltijos leidykla, 2000.- 188 p.
11. Blonskis J., Bukšnaitis V. ir kt. Programavimo C++ Builder pavyzdžiai. Kaunas: Smaltijos leidykla, 2004.- 362 p.
12. Adomavičius J., Dulinskienė T. Informatika 2. Uždavinių paruošimas sprendimui Matlab ir Mathcad aplinkose. Kaunas: Technologija, 2003.- 94 p.

7. SUMMARY

The Analysis of Creative Opportunities of Graphical User Interface Software of Computer Mathematics Systems

This paper is an analysis of creative opportunities of graphical user interface software of computer mathematics systems. There were two computer mathematics systems chosen: *Matlab* 7, *Maple* 10 and *Mathematica* 5.2 for this work. In order to compare the creative opportunities of graphical user interface software in computer mathematics systems and universal programming languages, *C++ Builder* 6 system was chosen.

In line, there were four application programs groups created in mathematics systems and in *C++ Builder* system. The process of creating these programs, the peculiarities of the codes and the final result were compared.

To sum up, computer mathematics systems may be used for creating application programs. Classical programming tasks may be implemented in these programs. Moreover, computer mathematics systems used for creating software cannot be changed by any other program that needs classical programming constructions, analytic computations and creating of graphical user interface.

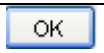
ANOTACIJA

Šis darbas analizuoja kompiuterinių matematikos sistemų (KMS) programų grafinės vartotojo sąsajos kūrimo galimybes. Tyrimui pasirinktos šios KMS: *Matlab* 7, *Maple* 10, *Mathematica* 5.2. Programų grafinės vartoto sąsajos kūrimo galimybėms KMS ir universaliose programavimo sistemose palyginti, pasirinkta *C++ Builder* 6 sistema.

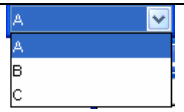
Tyrimui atlikti sukurtos keturios taikomųjų programų grupės kompiuterinėse matematikos sistemose ir *C++ Builder 6* sistemoje, palygintas kūrimo procesas, kodo ypatumai ir gautas rezultatas.

Iš atlikto darbo galima daryti išvadą, kad KMS gali būti panaudotos taikomosioms programoms kurti. Jose gali būti realizuojami klasikiniai programavimo uždaviniai. KMS yra nepakeičiamos sprendžiant uždavinius, kuriose reikia programuoti, naudoti matematinį aparatą bei sukurti patrauklią grafinę vartotojo sąsają.

1 PRIEDAS. PAGRINDINIAI KOMPONENTAI

C++ Builder 6	Maple 10	Matlab 7	Mathematica 5.2
 Button – mygtuko komponentas *Paskirtis* – reaguoti į įvykius programos vykdymo metu. Dažniausias įvykis – mygtuko spragtelėjimas. *Savybės*: Caption – simbolių eilutė (tekstas), kuri matoma komponento lange; Font – komponento teksto šriftas (dydis, stilius, spalva ir kt.); Enabled – nusako, ar komponentas reaguoja į įvykius; jei jo reikšmė yra *true*, mygtukas reaguoja į įvykius, jei *false* – nematomas; Visible – nusako, ar komponentas matomas; jei reikšmė yra true – matomas, jei *false* – nematomas.	 Button – mygtuko komponentas *Paskirtis* – reaguoti į įvykius programos vykdymo metu. Dažniausias įvykis – mygtuko paspaudimas. *Savybės*: background – mygtuko spalva; caption – tekstas ar simbolis, kuris atsiranda ant mygtuko; enabled – ar mygtukas aktyvus; jei nustatyta *false*, tai mygtukas atrodo neryškus ir jokie veiksmai, susiję su šiuo mygtuku negali būti atlikti; font – teksto šriftas; foreground – teksto spalva; image – paveikslas, pavaizduotas ant mygtuko; on click – veiksmas, kuris yra atliekamas, kai mygtukas paspaudžiamas; reference – nuoroda į mygtuko elementą; tooltip – tekstas, atsirandantis tooltip lange; visible – ar mygtukas yra matomas vartotojui; pagal	 Push Button – mygtuko komponentas *Paskirtis* – reaguoti į įvykius programos vykdymo metu. *Savybės*: BackgroundColor, BeingDeleted, BusyAction, ButtonDownFcn,CData, Callback, Clipping, CreateFcn, DeleteFcn, Enable, Extent, FontAngle, FontName, FontSize, FontUnits, FontWeight, ForegroundColor, HandleVisibility, HitTest, HorizontalAlignment, Interruptible, KeyPressFcn, ListboxTop, Max, Min, Position, SelectionHighlight, SliderStep, String, Style, Tag, TooltipString, UIContextMenu, Units, UserData, Value, Visible	 Button – mygtuko komponentas

	nutylėjimą reikšmė yra *true*.		
 CheckBox – veiksmo indikatoriaus komponentas *Paskirtis*. Tai indikatorius. Su kiekvienu spragtelėjimu jo būsena keičiasi: atsiranda arba išnyksta varnelė. Komponentas naudojamas tam, kad būtų įjungtos arba išjungtos tam tikros programos savybės. *Savybės*: Checked – reikšmė *true* rodo, kad indikatorius yra pažymėtas, reikšmė *false* – kad nepažymėtas; Caption – išsaugo užrašą šalia indikatoriaus.	 CheckBox – veiksmo pasirinkimo komponentas *Savybės*: background – komponento spalva; caption – tekstas, kuris atsiranda ant komponento; enabled - ar mygtukas aktyvus; jei nustatyta *false*, tai mygtukas atrodo neryškus ir jokie veiksmai, susiję su šiuo mygtuku negali būti atlikti; font – teksto šriftas; foreground – teksto spalva; image – paveikslas ar nuoroda į paveikslą; onchange – įvykis, kuris įvyksta, kai pasirenkamas vienas iš veiksmų ; reference – nuoroda į veiksmo pasirinkimo komponentą; tooltip – tekstas, atsirandantis užėjus su pele ant komponento; value – ar veiksmo pasirinkimo komponentas yra pažymėtas; pagal nutylėjimą, reikšmė yra false; visible – ar komponentas yra matomas; pagal nutylėjimą, reikšmė yra true;	 CheckBox – veiksmo indikatoriaus komponentas *Savybės*: BackgroundColor, BeingDeleted, BusyAction, ButtonDownFcn,CData, Callback, Clipping, CreateFcn, DeleteFcn, Enable, Extent, FontAngle, FontName, FontSize, FontUnits, FontWeight, ForegroundColor, HandleVisibility, HitTest, HorizontalAlignment, Interruptible, KeyPressFcn, ListboxTop, Max, Min, Position, SelectionHighlight, SliderStep, String, Style, Tag, TooltipString, UIContextMenu, Units, UserData, Value, Visible	 CheckBox – veiksmo indikatoriaus komponentas

 ComboBox – išskleidžiamojo sąrašo komponentas *Paskirtis* – saugoti išskleidžiamąjį sąrašą. Sąraše pele pažymėjus reikiamą eilutę, ji tampa aktyvi ir patalpinama lange. *Savybės*: Text – tai komponento lange matomas tekstas programos darbo pradžioje; Items – sąrašo elementai, matoti ComboBox lange; Items->Count – apibrėžia sąrašo eilučių skaičių; Items->Strings[i] – tai i-toji sąrašo eilutė ItemIndex – nusako aktyviosios eilutė, kuri matoma komponento langelyje numerį; Items->Add(E) – įtraukia į sąrašą naują eilutę E.	 ComboBox – išskleidžiamo sąrašo komponentas, vartotojas gali įvesti tekstą arba pasirinkti vieną įrašą iš sąrašo *Savybės*: background enabled font foreground halign – apibrėžia horizontalų lygiavimą komponento įrašams; height – aukštis pikseliais; pagal nutylėjimą aukštis atitinka bendrą visų įrašų aukštį; onchange reference tooltip valign – apibrėžia horizontalų lygiavimą komponento įrašams; value visible width – plotis pikseliais; pagal nutylėjimą plotis atitinka bendrą visų įrašų plotį;		 ComboBox – išskleidžiamo sąrašo komponentas
	 DropDownBox – išskleidžiamo sąrašo komponentas, vartotojas gali pasirinkti vieną įrašą iš sąrašo *Savybės* tokios pačios kaip ir ComboBox.		

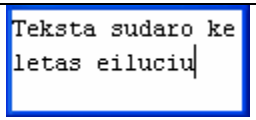
 Label – etiketės komponentas *Paskirtis* – parodyti įvairius užrašus formos lange. *Savybės*: Caption – simbolių eilutė, kuri matoma komponento lange; Font – nusako komponento šriftą;	 Label – etiketės komponentas *Savybės*: background, caption, enabled, font, foreground, halign, height, image, reference, tooltip, valign, visible, width.	 StaticText – teksto komponentas *Savybės*: BackgroundColor, BeingDeleted, BusyAction, ButtonDownFcn,CData, Callback, Clipping, CreateFcn, DeleteFcn, Enable, Extent, FontAngle, FontName, FontSize, FontUnits, FontWeight, ForegroundColor, HandleVisibility, HitTest, HorizontalAlignment, Interruptible, KeyPressFcn, ListboxTop, Max, Min, Position, SelectionHighlight, SliderStep, String, Style, Tag, TooltipString, UIContextMenu, Units, UserData, Value, Visible	 Label – etiketės komponentas
 ListBox – sąrašo komponentas *Paskirtis* – saugoti, rodyti ekrane sąrašo elementus. *Savybės*: Columns – (sąrašo elementų skaičius – 1); Items – sąrašo elementai (eilutės), matomi ListBox lange.	 ListBox – sąrašo komponentas, vartotojas gali pasirinkti vieną ar kelis įrašus *Savybės*: background, enabled, font, foreground, halign, onchange, reference, tooltip, valign, value, visible, width.	 ListBox – sąrašo komponentas *Savybės*: BackgroundColor, BeingDeleted, BusyAction, ButtonDownFcn,CData, Callback, Clipping, CreateFcn, DeleteFcn, Enable, Extent, FontAngle, FontName, FontSize, FontUnits, FontWeight, ForegroundColor, HandleVisibility, HitTest, HorizontalAlignment, Interruptible, KeyPressFcn, ListboxTop, Max, Min, Position,	 List – sąrašo komponentas

		SelectionHighlight, SliderStep, String, Style, Tag, TooltipString, UIContextMenu, Units, UserData, Value, Visible	
	 MathMLEditor – MathML redaktorius leidžia įvesti ir redaguoti matematines išraiškas *Savybės*: background, breakwidth – plotis pikseliais; focus – jei reikšmė yra *false*, tai vaizdas atrodo neryškus; height; outputformat – nurodo, ar bus pateikta pilna *MathML* prezentacija, ar tik turinys; palette – nurodo, pagrindinis ar išplėstas savybių sąrašas bus gaunamas spragtelėjus komponento srityje; visible; width; wrapped – nurodo, ar tekstas bus perkeliamas į kitą eilutę, kai neišsiteks *MathML* srityje;		
	 MathMlViewer – MathML vaizdavimo komponentas leidžia rodyti vaizdą 2-D *Savybės*:		

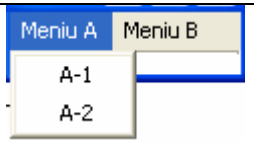
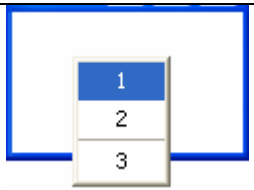
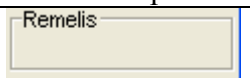
	background, breakwidth, fontsize, foreground, height, reference, value, visible, width, wrapped;		
 Chart – diagramų komponentas *Paskirtis* – piešti duomenų lentelių arba funkcijų diagramas ir grafikus. Reikia pasirinkti diagramos formą ir perduoti atvaizdavimui skirtus duomenis.	 Plotter – diagramų komponentas *Savybės*: background, height, reference, tooltip, value, visible, width;	 Axes – diagramų komponentas *Savybės*: ALim, ALimMode, ActivePositionProperty, AmbientLightColor, BeingDeleted, Box, BusyAction, ButtonDownFcn, CLim, CLimMode, CameraPosition, CameraPositionMode, CameraTarget, CameraTargetMode, CameraUpVector, CameraUpVectorMode, CameraViewAngle, CameraViewAngleMode, Clipping, Color, CreateFcn, CurrentPosition, DataAspectRatio, DataAspectRatioMode, DeleteFcn, DrawMode, FontAngle, FontName, FontSize, FontUnits, FontWeight, GridLineStyle, HandleVisibility, HitTest, Interruptible, Layer, LineStyleOrder, LineWidth, MinorGridLineStyle, NextPlot, OuterPosition, PlotBoxAspectRatio, PlotBoxAspectRatioMode, Position, Projection,	

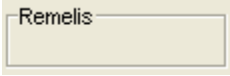
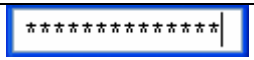
		SelectionHighlight, Tag, TickDir, TickDirMode, TickLength, TightInset, UIContextMenu, Units, UserData, View, Visible, XAxisLocation, XColor, XDir, XGrid, XLim, XLimMode, XMinorGrid, XMinorTick, XScale, XTick, XTickLabel, XTickLabelMode, XTickMode, YAxisLocation, YColor, YDir, YGrid, YLim, YLimMode, YMinorGrid, YMinorTick, YScale, YTick, YTickLabel, YTickLabelMode, YTickMode, ZColor, ZDir, ZGrid, ZLim, ZLimMode, ZMinorGrid, ZMinorTick, ZScale, ZTick, ZTickLabel, ZTickLabelMode, ZTickMode	
 RadioButton – alternatyvaus veiksmo indikatoriaus komponentas *Paskirtis* – tai indikatorius, skirtas veiksmui žymėti. Visada vartojama po kelis tokius mygtukus, kurie sudaro grupę. Pažymėti galima tik vieną iš jų. *Savybės*: Checked – reikšmė *true* rodo, kad indikatorius yra pažymėtas, reikšmė *false* – kad nepažymėtas;	 RadioButton – alternatyvaus veiksmo pasirinkimo komponentas *Savybės*: background, caption, enabled, font, foreground, group, image, reference, tooltip, value, visible;	 RadioButton – alternatyvaus veiksmo indikatorius	 RadioButton – alternatyvaus veiksmo indikatorius

Caption – išsaugo užrašą šalia indikatoriaus.			
 ScrollBar – slankiklis	 Slider – slankiklis *Savybės*: background; enabled; filled – nurodo, ar yra rodomas komponento takelis; foreground; lower – žemiausia komponento reikšmė; pagal nutylėjimą 0; majorticks – nurodo, kas kiek brūkšnelių bus sunumeruota; minorticks – nurodo nenumeruotus brūkšnelius; onchange; orientation – nurodo, ar komponentas stovės horizontaliai, ar vertikaliai; pagal nutylėjimą – horizontaliai; reference; showlabels – nurodo, ar numeruotų brūkšnelių etiketės matysis; showticks – nurodo, ar matysis brūkšneliai; snapticks – nurodo, ar spragtelės nuspaudus ant brūkšnelio su pele; tooltip; upper – didžiausia komponento reikšmė; pagal nutylėjimą ji yra 10; value – komponeto reikšmė; pagal	 Slider – slankiklis *Savybės*: BackgroundColor, BeingDeleted, BusyAction, ButtonDownFcn,CData, Callback, Clipping, CreateFcn, DeleteFcn, Enable, Extent, FontAngle, FontName, FontSize, FontUnits, FontWeight, ForegroundColor, HandleVisibility, HitTest, HorizontalAlignment, Interruptible, KeyPressFcn, ListboxTop, Max, Min, Position, SelectionHighlight, SliderStep, String, Style, Tag, TooltipString, UIContextMenu, Units, UserData, Value, Visible	 Slider - slankiklis

	nutylėjimą ji ya 0; visible;		
 Table – lentelės komponentas	 Table – lentelės komponentas *Savybės*: background; font; foreground; height; reference; tooltip; visible; width;		 Table – lentelės komponentas
 Edit – redagavimo komponentas *Paskirtis* – išsaugoti tekstą, kurį vartotojas užrašo šio komponento lange. Įvedimo metu tekstas gali būti redaguojamas. *Savybės*: Text – simbolių eilutė (tekstas), kuri matoma komponento lange. Programos vykdymo metu įvestas ar pakeistas tekstas tampa nauja Text savybės reikšme.	 TextField – teksto eilutės komponentas, jį sudaro tik viena eilutė iš Text Box komponento *Savybės*: background; cursor; editable; enabled; focus; font; foreground; halign; onchange; popupmenu; reference; tooltip; value; visible; width;	 EditText – redagavimo komponentas	 TextField – redagavimo komponentas
 Memo – teksto išvedimo komponentas	 TextBox – teksto eilučių komponentas, skirtas informacijos įvedimui ir išvedimui *Savybės*: background; cursor – kursoriaus vieta komponente;		 TextArea – kelių teksto eilučių išvedimo komponentas

	editable – nurodo, ar komponentas gali būti redaguojamas; enabled; focus; font; foreground; height; onchange; popupmenu – PopupMenu komponentas ar nuoroda į jį; reference; tooltip; visible; width; wrapped;		
	 ToggleButton – nuspaustas mygtukas *Savybės:* background – išjungto mygtuko spalva; caption – tekstas ar simbolis, kuris atsiranda ant mygtuko; enabled – ar mygtukas aktyvus; jei nustatyta *false*, tai mygtukas atrodo neryškus ir jokie veiksmai, susiję su šiuo mygtuku negali būti atlikti; font – teksto šriftas; foreground – teksto spalva; group – kai ši savybė pasirinkta, mygtuko elgesys yra panašus į RadioButton komponento: tik vienas mygtukas gali būti pasirinktas; kai savybė pasirinkta, mygtukai gali būti perjungti nepriklausomai vienas nuo kito; image – paveikslas, pavaizduotas ant mygtuko;	 ToggleButton – nuspaustas mygtukas	

	onchange – veiksmas, atliekamas paspaudus mygtuką; reference – nuoroda į mygtuko elementą; tooltip – tekstas, atsirandantis tooltip lange; value – mygtuko reikšmė; pagal nutylėjimą, reikšmė yra *false*; visible – ar mygtukas yra matomas vartotojui; pagal nutylėjimą reikšmė yra *true*.		
MainMenu – hierarchinio meniu komponentas *Paskirtis* – projektuoti programos meniu. Šis komponentas priklauso nematomų komponentų grupei, todėl jis gali būti padėtas bet kurioje formos lango vietoje.			 Menu – hierarchinio meniu komponentas
PopupMenu – kontekstinio meniu komponentas		Pop-upMenu – kontekstinio meniu komponentas	 PopupMenu – kontekstinio meniu komponentas
			 Frame – rėmelis *Paskirtis* – suformuoti rėmelį. Šis komponentas gali būti tik vienas grafinėje vartotojo

			sąsajoje.
 Panel – rėmelis		 Panel – rėmelis	 Panel – rėmelis *Paskirtis* – suformuoti rėmelį tam tikrai komponentų grupei. Šių komponentų gali būti keletas grafinėje vartotojo sąsajoje.
 GroupBox – grupavimo komponentas *Paskirtis* – suformuoti komponentų grupę. Tai tuščias skydelio tipo komponentas, skirtas valdantiems komponentams grupuoti, tokiems kaip RadioButton arba CheckBox. Pagrindinė *savybė* – Caption – užrašas kairiajame viršutiniame kampe.		 ButtonGroup – mygtukų grupavimo komponentas	
			 PasswordField – slaptažodžio įvedimo laukelis
		 ActiveX Control	

2 PRIEDAS. PROGRAMŲ TEKSTAI

1 uždavinys

Programos kodas Matlab kalba:

```
function varargout = skaiciuoklis(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @skaiciuoklis_OpeningFcn, ...
                   'gui_OutputFcn',  @skaiciuoklis_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before skaiciuoklis is made visible.
function skaiciuoklis_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = skaiciuoklis_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit2_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
x = str2double(get(handles.edit1,'string'));
y = str2double(get(handles.edit2,'string'));
ats = x + y;
set(handles.text2, 'String', ats);
set(handles.text1, 'String', '+');

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
x = str2double(get(handles.edit1,'string'));
y = str2double(get(handles.edit2,'string'));
ats = x - y;
set(handles.text2, 'String', ats);
set(handles.text1, 'String', '-');

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
x = str2double(get(handles.edit1,'string'));
y = str2double(get(handles.edit2,'string'));
ats = x * y;
set(handles.text2, 'String', ats);
set(handles.text1, 'String', '*');

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
x = str2double(get(handles.edit1,'string'));
y = str2double(get(handles.edit2,'string'));
ats = x / y;
set(handles.text2, 'String', ats);
set(handles.text1, 'String', '/');
```

Programos kodas Maple kalba: -

Programos kodas Mathematica kalba:

```
Needs["GUIKit`"]
Skaiciuoklis=
```

```
  Widget["Panel",{
    Widget["Label",{"text"->"Iveskite skaicius ir paspauskite reikiama mygtuka"}],
    {Widget["TextField",
       {"text"->"20"},Name->"laukas1"],
     Widget["Label",{"text"->"?"},Name→etikete1],
     Widget["TextField",
       {"text"->"5"},Name->"laukas2"]},
    {Widget["Button",{"text"->"+",
        BindEvent["action",Script[sudetis[]]]}],
     Widget["Button",{"text"->"-",
        BindEvent["action",Script[atimtis[]]]}],
     Widget["Button",{"text"->"*",
        BindEvent["action",Script[daugyba[]]]}],
     Widget["Button",{"text"->"/",
        BindEvent["action",Script[dalyba[]]]}],
     },
    {Widget["Label",{"text"→"Rezultatas = "},Name→etikete2],
     Widget["Label",{"text"→""},Name→etikete3]},
    Script[sudetis[]:=Module[{x,y,z},
        x=ToExpression[PropertyValue[{"laukas1","text"}]];
        y=ToExpression[PropertyValue[{"laukas2","text"}]];
        z=x+y;
        SetPropertyValue[{"etikete3","text"}, ToString[z, InputForm]];
        SetPropertyValue[{"etikete1","text"},"+"]
        ]],
    Script[atimtis[]:=Module[{x,y,z},
        x=ToExpression[PropertyValue[{"laukas1","text"}]];
        y=ToExpression[PropertyValue[{"laukas2","text"}]];
        z=x-y;
        SetPropertyValue[{"etikete3","text"}, ToString[z, InputForm]];
        SetPropertyValue[{"etikete1","text"},"-"]
        ]],
    Script[daugyba[]:=Module[{x,y,z},
        x=ToExpression[PropertyValue[{"laukas1","text"}]];
        y=ToExpression[PropertyValue[{"laukas2","text"}]];
        z=x*y;
        SetPropertyValue[{"etikete3","text"}, ToString[z, InputForm]];
        SetPropertyValue[{"etikete1","text"},"*"]
        ]],
    Script[dalyba[]:=Module[{x,y,z},
        x=ToExpression[PropertyValue[{"laukas1","text"}]];
        y=ToExpression[PropertyValue[{"laukas2","text"}]];
        z=x/y;
        SetPropertyValue[{"etikete3","text"}, ToString[N[z], InputForm]];
        SetPropertyValue[{"etikete1","text"},"/"]
        ]]
    }];
GUIRun[Skaiciuoklis];
```

Programos kodas C++ kalba:

Unit1.h:

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:   // IDE-managed Components
        TEdit *Edit1;
        TLabel *Label1;
        TEdit *Edit2;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
private:
        float ats;
public:            // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Unit1.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
```

```
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ Label1->Caption = "+";
  ats = StrToFloat(Edit1->Text) + StrToFloat(Edit2->Text);
  Label3->Caption = FloatToStrF(ats, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{ Label1->Caption = "-";
  ats = StrToFloat(Edit1->Text) - StrToFloat(Edit2->Text);
  Label3->Caption = FloatToStrF(ats, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{ Label1->Caption = "*";
  ats = StrToFloat(Edit1->Text) * StrToFloat(Edit2->Text);
  Label3->Caption = FloatToStrF(ats, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{ Label1->Caption = "/";
  ats = StrToFloat(Edit1->Text) / StrToFloat(Edit2->Text);
  Label3->Caption = FloatToStrF(ats, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
```

2 uždavinys

Programos kodas Matlab kalba:

```
function varargout = su_menu(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @su_menu_OpeningFcn, ...
```

```
                'gui_OutputFcn',  @su_menu_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before su_menu is made visible.
function su_menu_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for su_menu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = su_menu_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --------------------------------------------------------------------
function menu_failas_skaityti_Callback(hObject, eventdata, handles)
global A
FD = fopen('dv_mas.txt', 'r');
[m, count] = fscanf(FD, '%d\n', 1);
[n, count] = fscanf(FD, '%d\n', 1);
[A, count] = fscanf(FD, '%f', [m,n]);
fclose(FD);
set(handles.text2,'String',{'Pradine matrica:'});
set(handles.text3,'String',{num2str(A)});

% --------------------------------------------------------------------
function menu_failas_baigti_Callback(hObject, eventdata, handles)
close;

% --------------------------------------------------------------------
function menu_veiksmai_didziausias_Callback(hObject, eventdata, handles)
global A
did = max(max(A));
set(handles.text1, 'String', { 'Matricos didziausias elementas: ', did });

% --------------------------------------------------------------------
```

```
function menu_veiksmai_maziausias_Callback(hObject, eventdata, handles)
global A
maz = min(min(A));
set(handles.text1, 'String', { 'Matricos maziausias elementas: ', maz });

% --------------------------------------------------------------------
function menu_veiksmai_vidurkis_Callback(hObject, eventdata, handles)
global A
vid = mean(mean(A));
set(handles.text1, 'String', { 'Matricos elementu vidurkis: ', vid });

% --------------------------------------------------------------------
function failas_menu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function veiksmai_menu_Callback(hObject, eventdata, handles)
```

Programos kodas Maple kalba:

```
restart:
with(Maplets[Elements]):

Skaityti := proc()
  local fd,i ,j, nn, mm, g;
  global A, n, m, eil;
  fd := fopen("D:\\Magistrinis\\Maple_programos\\dv_mas.txt", READ);
  nn := fscanf(fd, "%d\n");
  n := nn[1];  #iš sąrašo elemento paverčia i skaičių
  mm := fscanf(fd, "%d\n");
  m := mm[1];
  A = array(1..n,1..m);
  for i from 1 to n do
    for j from 1 to m do
      g := fscanf(fd, "%f");
      A[i,j] := g[1];
    end do
  end do;
  fclose(fd);
  eil := "Pradinis masyvas";
  for i from 1 to n do
    for j from 1 to m do
      eil := cat(eil,A[i,j]);
    end do
  end do;
eil;
end proc:

Didziausias := proc()
  local i, j;
```

```
  global did;
  did := A[1,1];
  for i from 1 to n do
    for j from 1 to m do
      if  A[i,j]>did  then did := A[i,j] end if;
    end do;
  end do;
  did;
end proc:

Maziausias := proc()
  local i, j;
  global maz;
  maz := A[1,1];
  for i from 1 to n do
    for j from 1 to m do
      if  A[i,j]<maz  then maz := A[i,j] end if;
    end do;
  end do;
  maz;
end proc:

Vidurkis := proc()
  local i, j, sum;
  global vid;
  sum := 0;
  for i from 1 to n do
    for j from 1 to m do
      sum := sum + A[i,j];
    end do;
  end do;
  vid := sum/(n*m);
end proc:

Maplets[Display](Maplet(
   Window('title'="Dvimatis masyvas",
'menubar'='MB1',[TextBox['TB1'](10..40),TextBox['TB2'](2..40)]),
   MenuBar['MB1'](
       Menu("Failas",
           MenuItem("Skaityti",Evaluate('TB1' = "Skaityti")),
           MenuItem("Baigti", Shutdown())
       ),
       Menu("Veiksmai",
           MenuItem("Rasti didziausia", Evaluate('TB2'="Didziausias")),
           MenuItem("Rasti maziausia", Evaluate('TB2'="Maziausias")),
           MenuItem("Rasti vidurki", Evaluate('TB2'="Vidurkis"))
       )
   )
));
```

Programos kodas Mathematica kalba:

```
Needs["GUIKit`"]
Su_menu =GUIRun[
    Widget["Frame",{
       "menus"→
        Widget["MenuBar",{
           Widget["Menu",{"text"->"Failas",
              Widget["MenuItem",{"text"->"Skaityti",BindEvent["action",Script[Skaitymas[]]]}],
              Widget["MenuSeparator"],
              Widget["MenuItem",{"text"->"Baigti",BindEvent["action",Script[Baigimas[]]]}],
              }],
           Widget["Menu",{"text"→"Veiksmai",
              Widget["MenuItem",{"text"->"Rasti
didziausia",BindEvent["action",Script[Didziausias[]]]}],
              Widget["MenuItem",{"text"->"Rasti
maziausia",BindEvent["action",Script[Maziausias[]]]}],
              Widget["MenuItem",{"text"->"Rasti vidurki",BindEvent["action",Script[Vidurkis[]]]}]
              }],
           }],
       Widget["TextArea",{"rows"→8,"columns"→40,"lineWrap"→True},Name→laukas1],
       Widget["TextArea",{"rows"→2,"columns"→40,"lineWrap"→True},Name→laukas2],
       Script[Skaitymas[] := Module[{fd},
           fd = OpenRead["d:\Magistrinis\mathematica_programos\duom.txt"];
           n = Read[fd,Number];
           m = Read[fd,Number];
           Array[A,{n,m}];
           For[i=1,i≤n,i++,
             For[j=1,j≤m,j++,A[i,j] = Read[fd,Number]]];
           SetPropertyValue[{"laukas1","text"},  ToString[Array[A,{n,m}],InputForm]];
           Close[fd];
           ]],
       Script[Didziausias[] := Module[{did, dideil},
           did = A[1,1];
           For[i=1,i≤n,i++,
             For[j=1,j≤m,j++,
               If[A[i,j]>did,did=A[i,j]]]];
           dideil = ToString[" - didziausias elementas " did]; (*rezult eilutei suform*)
           SetPropertyValue[{"laukas2","text"},  ToString[dideil,InputForm]];
           ]],
       Script[Maziausias[] := Module[{maz},
           maz = A[1,1];
           For[i=1,i≤n,i++,
             For[j=1,j≤m,j++,
               If[A[i,j]<maz,maz=A[i,j]]]];
           mazeil =  ToString[" - maziausias elementas " maz];
           SetPropertyValue[{"laukas2","text"},  ToString[mazeil,InputForm]];
```

```
        ]],
    Script[Vidurkis[] := Module[{vid, videil,suma},
        suma = 0;
        For[i=1,i≤n,i++,
          For[j=1,j≤m,j++,
            suma = suma+A[i,j]]];
        vid = suma/(n*m);
        vid = N[vid];  (* trupmena pavercia realiuoju skaiciu *)
        videil = ToString[  " - vidurkis " vid];
        SetPropertyValue[{"laukas2","text"},  ToString[videil,InputForm]];
        ]],
    Script[Baigimas[] := Exit[]]
    }]
  ];
```

Programos kodas C++ kalba:

Unit1.h:

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <stdio.h>
#include <Grids.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:   // IDE-managed Components
    TMainMenu *MainMenu1;
    TMenuItem *Byla1;
    TMenuItem *Skaityti1;
    TMenuItem *Baigti1;
    TMenuItem *Veiksmai1;
    TMenuItem *Rastididziausia1;
    TMenuItem *Rastimaziausia1;
    TMenuItem *Rastividurki1;
    TLabel *Label1;
    TStringGrid *StringGrid1;
    TLabel *Label2;
    TLabel *Label3;
    void __fastcall Skaityti1Click(TObject *Sender);
    void __fastcall Rastididziausia1Click(TObject *Sender);
    void __fastcall Rastimaziausia1Click(TObject *Sender);
```

```
        void __fastcall Rastividurki1Click(TObject *Sender);
private:
        float A[10][10];
        int n, m;
        float max, min, vid;
        void Rodyti();
public:             // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Unit1.cpp:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Skaityti1Click(TObject *Sender)
{ int i, j;
  FILE *FD;
  FD = fopen("duom.txt", "r");
  fscanf(FD, "%d%d\n", &n, &m);
  for (i = 0; i<n; i++)
     for (j=0; j<m; j++)
        fscanf(FD, "%f", &A[i][j]);
  fclose(FD);
  Rodyti();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Rastididziausia1Click(TObject *Sender)
{ int i, j;
  max = A[0][0];
  for (i=0; i<n; i++)
     for (j=0; j<m; j++)
        if (A[i][j]>max) max = A[i][j];
  Label2->Caption = "Matricos didziausias elementas:";
```

```
  Label3->Caption = FloatToStrF(max, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Rastimaziausia1Click(TObject *Sender)
{ int i, j;
  min = A[0][0];
  for (i=0; i<n; i++)
     for (j=0; j<m; j++)
        if (A[i][j]<min) min = A[i][j];
  Label2->Caption = "Matricos maziausias elementas:";
  Label3->Caption = FloatToStrF(min, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Rastividurki1Click(TObject *Sender)
{ int i, j;
  float suma;
  for (i=0; i<n; i++)
     for (j=0; j<m; j++)
        suma = suma + A[i][j];
  vid = suma / (n*m);
  Label2->Caption = "Matricos elementu vidurkis:";
  Label3->Caption = FloatToStrF(vid, ffFixed, 7, 2);
}
//---------------------------------------------------------------------------
void TForm1::Rodyti()
{ int i, j;
  StringGrid1->ColCount = m+1;
  StringGrid1->RowCount = n+1;
  for (j=0; j<m; j++)
     StringGrid1->Cells[j+1][0] = "Stulp " + IntToStr(j);
  for (i=0; i<n; i++)
     StringGrid1->Cells[0][i+1] = "Eil " + IntToStr(i);
  for (i=0; i<n; i++)
     for (j=0; j<m; j++)
        StringGrid1->Cells[j+1][i+1] = FloatToStrF(A[i][j], ffFixed, 7, 2);
}
```

3 uždavinys

Programos kodas Matlab kalba:

```
function varargout = dif_int(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dif_int_OpeningFcn, ...
                   'gui_OutputFcn',  @dif_int_OutputFcn, ...
```

```
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Funkcija vykdoma prieš tai, kol sukuriama grafinė vartotojo sąsaja.
function dif_int_OpeningFcn(hObject, eventdata, handles, varargin)
global b
global c1
global c2

handles.output = hObject;
guidata(hObject, handles);
b = 0;
c1 = 0;
c2 = 1;
% --- Šios funkcijos rezultatai išvedami i komandų eilutę.
function varargout = dif_int_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)

% --- Funkcija vykdoma kuriant objektą, po to kai nustatytos savybes.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit2_Callback(hObject, eventdata, handles)

% --- Funkcija vykdoma kuriant objektą, po to kai nustatytos savybes.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Funkcija vykdoma, spragtelėjus pele veiksmo pasirinkimo komponentą.
```

```
function checkbox1_Callback(hObject, eventdata, handles)
global c1
if (get(hObject,'Value') == get(hObject,'Max'))
   c1 = 1;
else
   c1 = 0;
end

% --- Funkcija vykdoma, spragtelėjus pele veiksmo pasirinkimo komponentą.
function checkbox2_Callback(hObject, eventdata, handles)
global c2
if (get(hObject,'Value') == get(hObject,'Max'))
   c2 = 1;
else
   c2 = 0;
end

% --- Funkcija vykdoma pakeitus slankinklio vietą.
function slider1_Callback(hObject, eventdata, handles)

% --- Funkcija vykdoma kuriant objekta, po to kai nustatytos savybes.
function slider1_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Funkcija vykdoma paspaudus mygtuką Veiksmai.
function pushbutton1_Callback(hObject, eventdata, handles)
global b
global c1
global c2

kint = get(handles.edit2,'string');
k = sym(kint);
funkc = get(handles.edit1,'string');
if (b==0)
    rez = diff(funkc,k);
    rezult = char(rez);
else
   rez = int(funkc,k);
   rezult = char(rez);
end;
set(handles.edit3,'String',rezult);
% kuriamas grafikas
iki = get(handles.slider1,'Value');
if (c1==1)&(c2==1)
```

```
    axes(handles.axes1); %Pasirenkamos tinkamos asys
    ezplot(funkc,[0;iki]);
    hold on;
    %set(0,'Type','line','Color','red');
    %line('Color','r');
    ezplot(rezult,[0;iki]);
    hold off;
    set(handles.axes1,'XMinorTick','on');
    grid on;
elseif (c1==1)&(c2==0)
    axes(handles.axes1); % Pasirenkamos tinkamos ašys
    ezplot(funkc,[0;iki]);
    set(handles.axes1,'XMinorTick','on');
    grid on;
else (c1==0)&(c2==1)
    axes(handles.axes1); % Pasirenkamos tinkamos ašys
    ezplot(rezult,[0;iki]);
    set(handles.axes1,'XMinorTick','on');
    grid on;
end;

function edit3_Callback(hObject, eventdata, handles)

% --- Funkcija vykdoma kuriant objektą, po to kai nustatytos savybės.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Funkcija vykdoma, spragtelėjus pele veiksmo pasirinkimo komponentą.
function radiobutton1_Callback(hObject, eventdata, handles)
global b

b = 0;
if (get(hObject,'Value') == get(hObject,'Max'))
    set(handles.text2, 'String', 'diferencijuoti pagal kintamaji')
else
    set(handles.text2, 'String', 'integruoti pagal kintamaji')
end

% --- Funkcija vykdoma, spragtelėjus pele veiksmo pasirinkimo komponentą.
function radiobutton2_Callback(hObject, eventdata, handles)
global b

b = 1;
if (get(hObject,'Value') == get(hObject,'Max'))
    set(handles.text2, 'String',  'integruoti pagal kintamaji')
```

```
else
   set(handles.text2, 'String','diferencijuoti pagal kintamaji')
end

% --- Funkcija vykdoma paspaudus mygtuką Baigti
function pushbutton2_Callback(hObject, eventdata, handles)
Close;
```

Programos kodas Maple kalba:

```
restart:
with(Maplets[Elements]):

Integravimas := proc(f,g)
  local reiks, rez;
  reiks:=Maplets:-Tools:-Get('RB1');
  if (reiks='true')
      then rez:=diff(f,g)
      else rez:=int(f,g)
  end if;
end proc:

Grafikas := proc(f,g,h,k)
  local pas1, pas2;
  pas1:=Maplets:-Tools:-Get('ChB1');
  pas2:=Maplets:-Tools:-Get('ChB2');
  if (pas1 = 'true') and (pas2 = 'true')
    then plot([f, h], g=0..k)
    else if (pas1 = 'true')
            then plot(f, g=0..k)
            else plot(h, g=0..k);
         end if;
  end if;
end proc:

Dif_int := Maplet( Window( 'title'="Matematinis pavyzdys", BoxColumn('vscroll'='always',
   [BoxColumn(border=true,
   RadioButton['RB1']("diferencijuoti", 'value'='true',
'group'='BG1',Action(SetOption('L2'('caption')="diferencijuoti pagal kintamaji"),
SetOption('ChB2'('caption')="rodyti diferencijuotos funkcijos grafika"))),
   RadioButton['RB2']("integruoti     ", 'value'='false',
'group'='BG1',Action(SetOption('L2'('caption')="integruoti pagal kintamaji"),
SetOption('ChB2'('caption')="rodyti integruotos funkcijos grafika     "))))),
   BoxColumn(border=true,
   [Label['L1']( "Iveskite funkcija:            "), TextField['TF1']("x^5")],
   [Label['L2']("diferencijuoti pagal kintamaji: "), TextField['TF2']("x",3)])],
   BoxColumn(border=true,halign=left,
   [CheckBox['ChB1'](caption="rodyti funkcijos grafika                    ",'value'=false)],
   [CheckBox['ChB2'](caption="rodyti diferencijuotos funkcijos grafika",'value'=true)]
```

```
   ),
   Slider['SL1']( 0..20, 5, 'showticks', 'majorticks'=5, 'minorticks'=1, 'visible'='true',   Evaluate  (
'PL1' = 'plot([TF1, TB1], TF2=0..SL1)' ) ),
   TextBox['TB1']( 'editable' = 'false', 3..40 ),
   [Button("Veiksmai", Action(Evaluate('TB1' = 'Integravimas(TF1,TF2)'),Evaluate( 'PL1' =
'Grafikas(TF1,TF2,TB1,SL1)' ))),
    Button("Baigti", Shutdown())],
    Plotter['PL1']()
) ),
    ButtonGroup['BG1']()
     ):
Maplets[Display](Dif_int);
```

Programos kodas Mathematica kalba:

```
Needs["GUIKit`"]
ref = GUIRun[
     Widget["Panel", {
        {WidgetGroup[{Widget["RadioButton",{"text" → "diferencijuoti",
              "selected" → True, BindEvent["action",Script[pakeisti1[]]]}, Name->"rb1"],
            Widget["RadioButton",{"text" → "integruoti",
              "selected" → False, BindEvent["action", Script[pakeisti2[]]]}, Name->"rb2"]},
WidgetLayout→{"Border"→{{{15,15},{10,15}},"Pasirinkite",{{15,15},{10,15}}}}],
          WidgetGroup[{{Widget["Label",{"text"->"Iveskite funkcija"}],
              Widget["TextField",{"text"→"x^5","columns"→10},Name→"tf1"]},
            {Widget["Label",{"text"->"diferencijuoti pagal"},Name→"l2"],
              Widget["TextField",{"text"→"x","columns"→10},Name→"tf2"]}}, WidgetLayout->
{"Border"→{{{20,25},{10,15}},"Iveskite",{{20,25},{10,15}}}}]
         },
        WidgetGroup[{{Widget["CheckBox",{"text"->"rodyti funkcijos
grafika",BindEvent["action",Script[graf[]]]},Name→"cb1"],
            Widget["CheckBox",{"text"->"rodyti diferencijuotos funkcijos grafika","selected"→True,
BindEvent["action",Script[grafikasint[]]]},Name→"cb2"]}}, WidgetLayout->
{"Border"→{{{20,20},{10,20}},"Pasirinkite",{{20,20},{10,20}}}}],
        Widget["Slider",{
           BindEvent["change",
             Script[slinkti[];]]},Name→"slider1",
          WidgetLayout→{"Stretching"→{Maximize, False}}],
        {WidgetFill[],
          Widget["Button",{"text"->"Veiksmai",BindEvent["action",Script[mygt[]]]}],
          Widget["Button",{"text"->"Baigti",BindEvent["action",Script[uzd[]]]}]},
        {Widget["Label",{"text"->"Rezultatas"}],
          Widget["TextField",{"text"->"","editable"→"False"},Name→"tf3"]},
        Widget["MathPanel",{"preferredSize"→Widget["Dimension",{"width"→288, "height"
→288}]},Name→"canvas",
          WidgetLayout→{"Stretching"→{True, Maximize}}],
        Widget["ButtonGroup",{
           WidgetReference["rb1"],
```

```
        WidgetReference["rb2"]}],
      Script[pakeisti1[]:=
        (If[PropertyValue[{"rb1","selected"}], SetPropertyValue[{"l2", "text"},"diferencijuoti
pagal"];
          SetPropertyValue[{"cb2", "text"},"rodyti diferencijuotos funkcijos grafika"]])
       ],
      Script[pakeisti2[]:=
        (If[PropertyValue[{"rb2","selected"}],SetPropertyValue[{"l2","text"},"integruoti pagal"];
          SetPropertyValue[{"cb2", "text"},"rodyti suintegruotos funkcijos grafika"]])
       ],
      Script[mygt[]:=Block[{funkc,kint,iki,$DisplayFunction=Identity},
          funkc=ToExpression[PropertyValue[{"tf1","text"}]];
          kint=ToExpression[PropertyValue[{"tf2","text"}]];
          dif=Dt[funkc,kint];
          int=Integrate[funkc,kint];
          iki=ToExpression[PropertyValue[{"slider1","value"}]];

(If[PropertyValue[{"rb1","selected"}],SetPropertyValue[{"tf3","text"},ToString[dif,InputForm]]]);
(If[PropertyValue[{"rb2","selected"}],SetPropertyValue[{"tf3","text"},ToString[int,InputForm]]]);
          (Which[
(PropertyValue[{"cb1","selected"}])&&(PropertyValue[{"cb2","selected"}])&&(PropertyValue[{"r
b1","selected"}]), SetPropertyValue[{"canvas","mathCommand"}, ToString[Plot[
                {funkc,dif},{x,0,iki}],InputForm]],
(PropertyValue[{"cb1","selected"}])&&(PropertyValue[{"cb2","selected"}])&&(PropertyValue[{"r
b2","selected"}]), SetPropertyValue[{"canvas","mathCommand"}, ToString[Plot[
                {funkc,int},{x,0,iki}],InputForm]],
            (PropertyValue[{"cb2","selected"}])&&(PropertyValue[{"rb1","selected"}]),
            SetPropertyValue[{"canvas","mathCommand"},
ToString[Plot[dif,{x,0,iki},PlotRange→All],InputForm]],
(PropertyValue[{"cb2","selected"}])&&(PropertyValue[{"rb2","selected"}]),
            SetPropertyValue[{"canvas","mathCommand"},
ToString[Plot[int,{x,0,iki},PlotRange→All],InputForm]],
            PropertyValue[{"cb1","selected"}],
            SetPropertyValue[{"canvas","mathCommand"},
ToString[Plot[funkc,{x,0,iki},PlotRange→All],InputForm]]
            ])
          ]],
      Script[uzd[]:=CloseGUIObject[ref]]
      }]
    ];
```

4 uždavinys

Programos kodas C++ kalba:

Unit1.h:

```
//---------------------------------------------------------------------------
```

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------
const CKiek = 100;
struct TStudentas
{ char pav[15],         //pavarde
      vard[15];        //vardas
  float vid;           //vidurkis
};
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:   // IDE-managed Components
       TLabel *Label1;
       TMemo *Memo1;
       TMainMenu *MainMenu1;
       TMenuItem *Byla1;
       TMenuItem *Skaityti1;
       TMenuItem *Baigti1;
       TMenuItem *Veiksmai1;
       TMenuItem *Surikiuotipagalvidurki1;
       TMenuItem *Spausdinti1;
       void __fastcall Skaityti1Click(TObject *Sender);
       void __fastcall Baigti1Click(TObject *Sender);
       void __fastcall Surikiuotipagalvidurki1Click(TObject *Sender);
       void __fastcall Spausdinti1Click(TObject *Sender);
private:
  TStudentas A[CKiek];
  int n;
  void Rodyti();
  public:                    // User declarations
       __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Unit1.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
```

```
#pragma hdrstop

#include "Unit1.h"
#include <stdio.h>
#include <string.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void TForm1::Rodyti()
{ AnsiString E; int i;
  Label1->Caption = "Studentu sarasas";
  Memo1->Lines->LoadFromFile("sar1.txt");
  Memo1->Lines->Add("-------------------------------");
  Memo1->Lines->Add("STUDENTU VIDURKIAI");
  for (i=0; i<n; i++)
  { E = A[i].pav;
    Memo1->Lines->Add(E + " " + FloatToStrF(A[i].vid,ffFixed, 6, 2));
  }
  Memo1->Lines->Add("-------------------------------");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Skaityti1Click(TObject *Sender)
{ int i, k;  float sum;  int B[10];
  FILE *FD;
  FD = fopen("sar1.txt", "r");
  if (FD == NULL)
  { Memo1->Lines->Clear();
    Memo1->Lines->Add("Duomenu failas nesurastas");
  }
  n = 0;
  k = 3;
  while (!feof(FD) && (n<CKiek))
  { fscanf(FD, "%15s%15s", A[n].pav, A[n].vard);
    for (i=0; i<=k; i++)
      fscanf(FD, "%d", &B[i]);
    fscanf(FD, "\n");
    sum = 0;
    for (i=0; i<=k; i++)
      sum = sum + B[i];
    A[n].vid = sum/(k+1);
    n = n + 1;
```

```
  }
  fclose(FD);
  Rodyti();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Baigti1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Surikiuotipagalvidurki1Click(TObject *Sender)
{ int i, j; TStudentas C;
  for (i=0; i<n-1; i++)
    for (j=i+1; j<n; j++)
    { if (A[j].vid>A[i].vid)
      { C=A[i];
        A[i]=A[j];
        A[j]=C;
      }
    }
   Rodyti();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Spausdinti1Click(TObject *Sender)
{
 FILE *FR;  int i;
  FR = fopen("rez.txt", "w");
  for (i=0; i<n; i++)
  { fprintf(FR, "%15s%15s%7.2f\n", A[i].pav, A[i].vard, A[i].vid);
  }
  fclose(FR);
}
//---------------------------------------------------------------------------
```

Programos kodas Matlab kalba:

```
function varargout = irasai_menu(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @irasai_menu_OpeningFcn, ...
                   'gui_OutputFcn',  @irasai_menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --------------------------------------------------------------------
function irasai_menu_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = irasai_menu_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% --------------------------------------------------------------------
function menu_failas_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function menu_failas_skaityti_Callback(hObject, eventdata, handles)
global A
global n
FD = fopen('ir_duom.txt','r');
n = 1;
A = struct('vard',{},'pav',{},'vidu',{},'vid',{});
while feof(FD)==0
   [A(n).vard,count] = fscanf(FD,'%9s',1);
   [A(n).pav,count] = fscanf(FD,'%15s',1);
   [A(n).vidu,count] = fscanf(FD,'%f\n',4);
   A(n).vid = mean(A(n).vidu);
   n = n + 1;
end;
n = n - 1;
fclose(FD);
set(handles.text1,'String',{'Pradiniai duomenys:'});
eil = struct('visi',{});
for i = 1:n
  eil(i).visi = strcat(A(i).pav,' ', A(i).vard,' ', num2str(A(i).vid));
end;
set(handles.text2,'String',{eil.visi});

% --------------------------------------------------------------------
function menu_failas_baigti_Callback(hObject, eventdata, handles)
close;
```

```
% --------------------------------------------------------------------
function menu_veiksmai_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function menu_veiksmai_surikiuoti_Callback(hObject, eventdata, handles)
global A
global n
for i = 1:n-1
   for j = i+1:n
      if A(j).vid > A(i).vid
         C = A(i);
         A(i) = A(j);
         A(j) = C;
      end;
   end;
end;
set(handles.text1,'String',{'Surikiuotas irasu masyvas:'});
eil = struct('visi',{});
for i = 1:n
  eil(i).visi = strcat(A(i).pav,' ', A(i).vard,' ', num2str(A(i).vid));
end;
set(handles.text2,'String',{eil.visi});

% --------------------------------------------------------------------
function menu_veiksmai_spausdinti_Callback(hObject, eventdata, handles)
global A
global n
FR = fopen('ir_rez.txt','w');
fprintf(FR,'Surikiuotas irasu masyvas \n');
fprintf(FR,' \n');
for i = 1:n
   fprintf(FR,'%15s %15s %6.2f', A(i).pav, A(i).vard, A(i).vid);
   fprintf(FR, ' \n');
end;
fclose(FR);
```