ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

Andrius Baranauskas

AUTOMATIZUOTA TEORINIŲ TYRIMŲ TERPĖ NAUDOJANTI SIMBOLINIUS SKAIČIAVIMUS IR ŽINIŲ BAZES

Magistro darbas

Vadovas
doc. dr. G. Daunys

ŠIAULIAI, 2004

TURINYS

SUMARY

The main aims of automatized environment of theoretical researches are to help and accelerate complicated signals' processing, creation of algorithms of neurons' networks, filters and other symbolic and calculation engineering researches which can be understood and promoted by every person. The environment has been implemented using simple as possible interaction of machine and the human communication and the possibility to actuate into research bases of systems of knowledge.

ĮŽANGA

Pasaulis sparčiai keičiasi. Kaip matyti, strategijoje didelis dėmesys skiriamas žmogaus ir kompiuterio sąveikai. Švietimo specialistai pastebėjo, kad informacinės technologijos geriau ir greičiau padeda įsisavinti žinias bei išsiugdyti reikiamus įgūdžius. Darant pirmuosius Europos Sąjungos(ES) narystės žingsnius, Lietuvos universitetų absolventai turi išmokti ruošti projektus konkurencingai ES bei platesnėms rinkoms. Pastarieji metai pasižymi dideliu asmeninių kompiuterių paplitimu ir jų skaičiuojamosios galios žymiu išaugimu.

Šiuo metu, po daugiau kaip 30 metų, programinės įrangos krizė vis dar neišspręsta. Nors programavimo inžinerijos metoduose bei technologijose, sistemų vystymo priemonėse ir personalo meistriškume yra didelių pasiekimų, paklausa programinei įrangai auga greičiau, negu jos kūrimo produktyvumas[1].

Besikeičiančios žmogaus pagalbinės techninės priemonės turi atsispindėti studijų procese. Tai ypatingai liečia inžinerines studijas. Vis dažniau stengiamasi integruoti programinę įrangą į mokslinius tyrimus. Tai leidžia pasiekti geresnių, greitesnių ir kartais netikėtų rezultatų. Tačiau neretai tokia programinė įranga yra brangiai apmokama ir nespecializuota. Materialiosios studijų programų realizavimo sąlygos elektronikos srityje yra geros, tačiau nepakankamas finansavimas naujos technikos ir programinės įrangos įsigijimui skatina kurti savo,- plataus pritaikomumo ir labiau specializuotą programinę įrangą, kuri taptų pagalbiniu darbo įrankiu ne tik mokslinius tyrimus atliekantiems magistrantams, bet ir sudėtingus skaičiavimus atliekantiems elektronikos inžinerijos bakalaurams ar doktorantams. Tokia programine įranga dirbančiam inžinieriui, dažniausiai kyla novatoriškų idėjų, kurias “užmuša” galingi bendros paskirties didelių kompanijų programiniai paketai.

Daugelis vartotojų, vien perskaitę reikalavimų specifikaciją, sunkiai įsivaizduoja kaip sistema yra naudojama. Todėl vienas būdas pasipriešinti sunkumams, kuriuos turi vartotojai formuluojant ir suprantant statiškas specifikacijas, yra sukurti sistemos prototipą ir leisti sistemos vartotojams eksperimentuoti su juo. Taip pat, įvertinami sistemos reikalavimai, kadangi vartotojai randa reikalavimų klaidas ir trūkumus[2].

1. PROGRAMINĖS ĮRANGOS APŽVALGA IR PROBLEMOS FORMAVIMAS

Šiuo metu elektronikos specialybės dėstytojai ir studentai studijų procese, o inžinieriai darbui gali panaudoti įvairius siūlomus programinius paketus. Produktus galima suskirstyti į grupes:

1. *universalūs* kompiuterių algebros paketai;
2. *specializuoti priedai* universaliems paketams;
3. *specifinės paskirties* programinė įranga.

Kiekvienoje grupėje egzistuoja komercinė ir nemokamai platinama, dažniausiai universitetuose sukurta programinė įranga. Pirmos grupės komerciniai produktai gerai žinomi: Waterloo Maple[3] paketas Maple, Wolfram Research Inc.[4] Matematica, Sci Face Software GmbH & Co[5] Mupad, Mathworks[6] Symbolic Mathematics Toolbox for Matlab. Paminėtų ir kitų produktų sąrašas pateiktas organizacijos SymbolicNet interneto svetainėje[7]. Ten pat galima rasti ilgą sąrašą ir nemokamai platinamų bei atviro kodo simbolinių skaičiavimų paketų. Iš jų reikėtų išskirti projektą Yacas(Yet Another Computer Algebra System)[8], kuris pritaikytas kuriamoje automatizuotoje teorinių tyrimų terpėje. Jo pagrindą sukūrė olandas A.Pinkus C++ kalba. Šiuo metu prie jo tobulinimo prisijungė kiti mokslininkai ir programuotojai.

Elektronikos srityje simboliniai skaičiavimai dažniausiai naudojami schemų signalų analizei. Tam tinka universalieji paketai[9,10]. “Signalų ir sistemų” disciplinai paketus lengva pritaikyti, kadangi jie turi funkcijas, atliekančias tiesiogines ir atvirkštines Furje, Laplaso ir z transformacijas. I. Munteanu ir D. Ioan nagrinėjo, kaip Maple paketą panaudoti Elektromagnetizmo studijoms[11]. Šio straipsnio autoriai turi patirtį panaudojant Matlab paketo simbolinius skaičiavimus dėstant apie puslaidininkinius prietaisus ar analoginius filtrus.

Schemų analizę palengvina priedai, skirti universalioms programoms. Plačiai paplitęs priedas Mathematica paketui Analog Insydes[12], sukurtas Fraunhofer institute. Šis paketas brangus produktas. Jo profesinė licenzija kainuoja 5000 eurų, o akademinė – 1000. Be to reikalingas ir pats Mathematica paketas su papildomai perkamu Symbolic Mathematics Toolbox, kurio kaina akademinėms įstaigoms 400 eurų. “Georgia Institute of Technology” pasipiktinęs per griežta tokių programinių paketų licenzija. Tarkim įdiegus Mathematica paketą, jo šriftai gali būti naudojami tik matematinėms išraiškoms ir Mathematica komandoms rašyti.

1996 metais Huelsman atliktoje analizėje[13], surinkti duomenys apie to meto specialiuosius paketus, skirtus schemų analizei. Tai paketai: SC, PC-SNAPS, Sspice, SAPWIN, ISAAC, ASAP, SYNAP, SAPFC, SCYMBAL, SCAPP ir RANIER. Vieni iš šių paketų išliko ir tobulinami toliau. Pvz. Florencijos universitete sukurtas ir atnaujinamas SAPWIN paketas[14], kurio versijas galima nemokamai atsisiųsti internetu.

Siūlomi programiniai paketai ne visuomet yra priimtini. Vieni iš jų yra komerciniai ir būtina pirkti licenzijas. Kiti turi mažai galimybių. Todėl kuriama sava programinė įranga, kuri patenkintų mūsų lūkesčius ir keliamus reikalavimus.

2. NEMOKAMA PROGRAMINĖ ĮRANGA

Tikriausiai antra pagal populiarumą nemokamų prekių sritis – programinė įranga. Nemokamų lietuviškų programų surasti tikrai galima. Pirmiausia reiktų paminėti tinklapį http://programos.mes.lt, kur lankytojai ras vieną didžiausių lietuviškų programų pasirinkimą. Tiesa, atrodo, jog tinklapis merdėja, o dauguma programų „kabo“ – iš jų likęs tik pavadinimas, kurių šiame tinklapyje tikrai apstu. Vis dėlto šansų rasti norimą programą čia yra. Nemokamų įrankių galima parsisiųsti iš dar kelių svetainių (http://lietuviskassoftas.ten.lt/, http://apocalypse3000.20m.com/ ir kt.), tačiau pirmajai jos neprilygsta nei savo dizainu, nei programų pasiūla.

Lankytojai, kurie tikisi parsisiųsti nemokamų demonstracinių programų versijų iš jas kuriančių firmų, dažnai nusivils, nes anaiptol ne visos įmonės savo produkciją pateikia internete. Rasti, be abejo, galima, pavyzdžiui, apsilankę bendrovės „Infotema“ (http://www.infotema.lt/) svetainėje internautai gali parsisiųsti nemokamą apskaitos programos versiją.

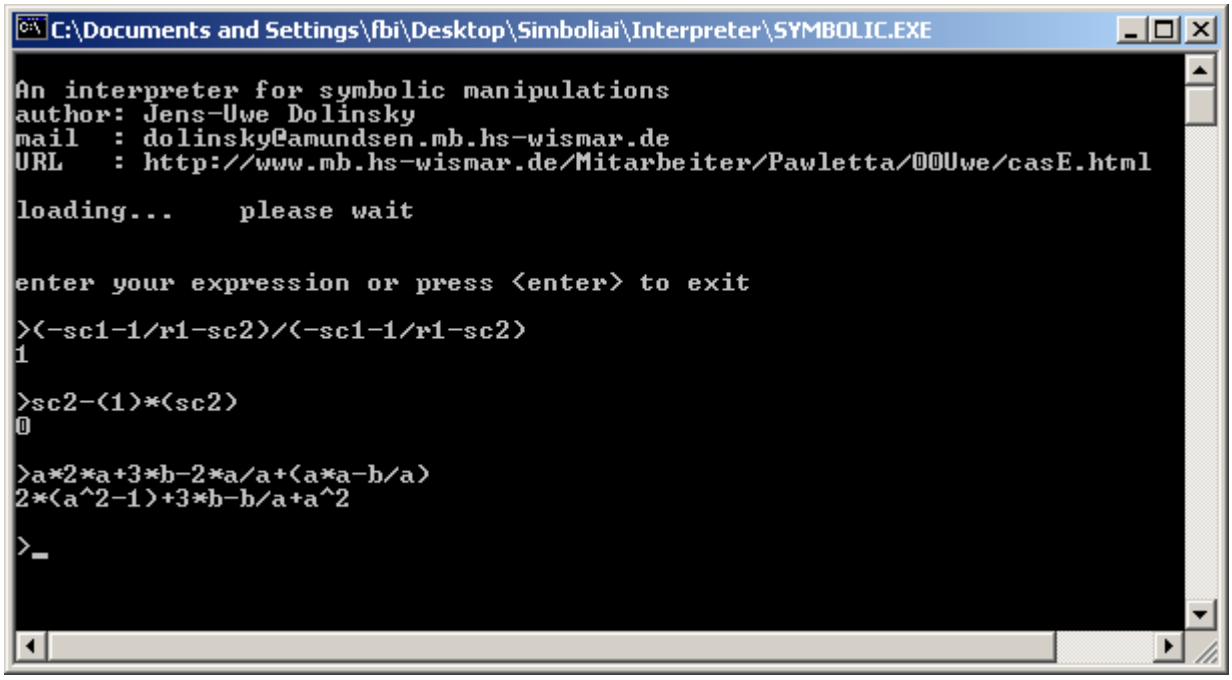
Iš to, ką radome, galima padaryti gana liūdną išvadą – nemokamų programų pasiūla lietuviškame internete akivaizdžiai mažėja. „Programos.mes.lt“ – merdėja, žinomo tinklapio „Programmers.lt“ domenas parduodamas, o kiti, mažesni, tinklapiai vargu ar patenkins internautų poreikius. Tad visai nenuostabu, kad dauguma vartotojų, užuot ieškoję programų lietuviškame internete, dažniausiai naudojasi paieškos sistema „Google“ (www.google.com) ar tinklapiu „Tucows“ (www.tucows.vu.lt)[15].

Vienas didžiausių pasaulinių nemokamos ir atviro kodo programinės įrangos tinklapių yra http://sourceforge.net . Pasidairėme ieškodami nemokamos simbolinių skaičiavimų ar panašios programinės įrangos.

Pavyko rasti pora nemokamai platinamų programų , tai “Interpreter” ir “Yacas”. Jas apžvelgsime 2.1 ir 2.2 skyriuose.

2.1 “Symbolic” bibliotekos pritaikymas ir apžvalga

“Simbolic.dll” ,- tai programinio paketo Interpreter(*Interpreter for symbolic manipulation of mathematical expressions*) simbolinių skaičiavimų prastinant reiškinius biblioteka. Šią biblioteką parašė Jens-Uwe Dolinsky iš Veimaro universiteto(Vokietija). Pats programinis Interpreter paketas veikia MS-DOS ir Linux consolėse(3.1.1pav.). Taip pat egzistuoja ir JAVA versija (Aplet).



2.1.1. pav. Interpreter MS-DOS aplinkoje

Pasak autoriaus, šio programinio paketo tikslas buvo automatinis manipuliavimas matematiniais reiškiniais analizuojant jų sintaksę ir semantiką bei prastinti atsižvelgiant į pagrindines algebros taisykles. Kitaip tariant Interpreter sudaro paprasta kompiuterių algebros sistema. Ateityje autorius numato simbolinę diferenciavimo ir integravimo galimybę. Pagrindinis dėmesys plėtojant šį paketą,- pasiekti lankstumą, funkcionalumą, naudojant mažiausią programų architektūrą ir paprastą prisitaikomumą prie daugelio skaičiavimo problemų.

Interpreter naudojimo galimybės:

• kaip komandinės eilutės aplikacija, kuri ima tekstą iš standartinio įvesties lauko ir spausdiną į standartinį išvesties lauką.

• kaip dinaminė biblioteka (DLL)

2.1.1 Sintaksė ir funkcijos

2.1.1.1 lentelė

	Sutartas žymėjimas	Paaiškinimas
operatoriai	`+,-,*,/,^`	Pagrindinės operacijos
konstantos	`PI`	Apskritimo ilgis
Pagrindinės f-jos	`SQRT(x)`	Šaknis iš x

	EXP(x)	eksponentinė funkcija
	LN(x)	Natūrinis logaritmas
	SIN(x),COS(x),TAN(x),COT(x)	Trigonometrinės f-jos
	ASIN(x),ACOS(x),ATAN(x)	Inversinės trigonometrinės f-jos
	SINH(x),COSH(x)	Hyperbolinės f-jos
	ABS(x)	Absoliuti x reikšmė
	FAK(x)	X faktorialas
Lygčių sprendim.	SOLVE(expr1=expr2,x)	Tiesinės ar kvadratinės lygtys expr1-expr2=0 sprendžiamos simboliškai per x.
diferenciavimas	DIFF(f,x)	F išvestinė atsižvelgiant į x
	DIFF(f,x,n)	n- os eilės f išvestinės atsižvelgiant į x
integravimas	INT(f,x)	Neapibrėžtas f integralas atsižvelgiant į x
	INT(f,x,i1,i2)	Apibrėžtas f integralas atsižvelgiant į x intervale i1 iki i2
Tayloro apriksimacija	TAYLOR(f,x,x0,n)	Teyloro aproksimacija f atsižvelgiant į x kai x=x0 n-tos eilės
Furje analizė	FOURIER(f,x,t1,t2,n)	Furje eilė f atsižvelgiant į x intervalui nuo t1 iki t2 n-tos eilės
Slenkančio kablelio skaič.	APPROX(x)	Konstantos x išraiška skaičiuoja 16 skaičių po kablelio.

Autorius nepateikia išeities kodo, todėl bibliotekoje nieko negalime keisti ir į simbolinius skaičiavimus integruojame standartinę jau sukompiliuotą “Simbolic.dll” biblioteką bei pasinaudojame jos “SOLVE()” funkcija(2.1.1.1pvz).

2.1.1.1 pavyzdys

```
solve(a*b*x+a*x+a*x+x=2,x)   # norime rasti x
x=2/((b+2)*a+1)
```

2.1.2 Trūkumai ir privalumai

Trūkumai:

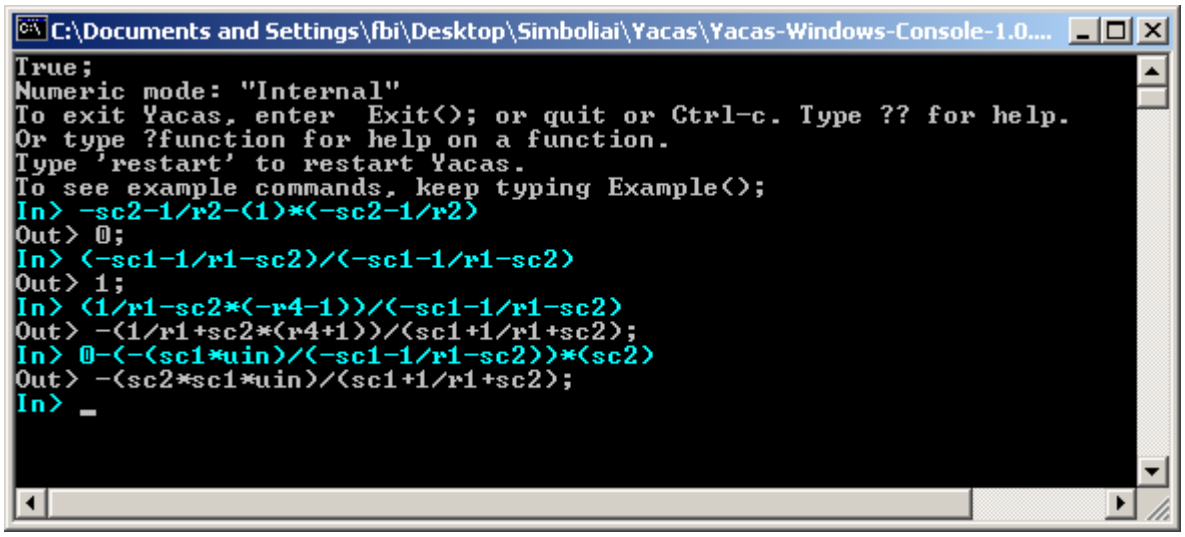
1) Funkcija gali išreikšti tik vieną nežinomąjį iš vienos tiesinės lygties.
2) Nesugeba apdoroti ilgų reiškinių.
3) Neskaičiuoja įvairaus tipo lygčių.
4) Nėra galimybės išsaugoti ir pakrauti rezultatus.

Privalumai:

1) Pakankamai supaprastina bendras reiškinių formas.
2) Lankstus sintaksės mechanizmas

2.2 "Yacas" bibliotekos pritaikymas ir apžvalga

"Yacas"(Yet Another Computer Algebra System) yra publikuotas 2000-siais metais Free Software Foundation, Inc.[16]. ir turi GNU Free Documentation License[17]. Taipogi randamas GNU/Linux "stable" programinės įrangos šaltiniuose[]. Tai reiškia, kad "Yacas" programinis paketas priklauso visuomenei ir autorius atsisakęs autorinių teisių į šią programinę įrangą. Tai leidžia kitiems naudotis ir keisti programos kodą pagal savo poreikius.



3.2.1. "Yacas" MS-DOS aplinkoje

Yacas – nedidelė, didelį lankstumą turinti kompiuterinės algebros sistema ir programavimo kalba, kuri labai panaši į visiems gerai žinomą C ,- įterpiant operatorių sintaksę. Programos distributyvą sudaro nedidelė matematinių funkcijų biblioteka, o pagrindinį "variklį" sudaro arbitriškai tiksli aritmetika(greitiems skaičiavimams atlikti).

Yacas naudojimo galimybės:

- kaip komandinės eilutės aplikacija, kuri ima tekstą iš standartinio įvesties lauko ir spausdiną į standartinį išvesties lauką.
- kaip tarnybinė stotis naudojanti tam skirtą prievadą.(t.y. nutolusiame kompiuteryje)
- kaip dinaminė biblioteka (DLL)

2.2.1 Sintaksė ir funkcijos

2.2.1.1 lentelė

	Sutartas žymėjimas	Paaiškinimas
Operatoriai	`+,-,*,/,^, %`	
Pagrindinės f-jos	`SIN(x),COS(x),TAN(x),COT(x)`	`Trigonometrinės f-jos`
	`ASIN(x),ACOS(x),ATAN(x)`	`Inversinės trigonometrinės f-jos`
	x!	`x faktorialas`

Reiškinių prastinimas	`Simplify(expr)`	`Reiškinio supaprastinta forma`
	`Solve(a+x*y==z,x);`	`Tiesinės ir kvadratinės lygtys`
	`PrettyForm(%);`	`Pateikimas laužytoje formoje`
	`Set(a,Cos(0));`	`Set tolygus operatoriui ":="`
	`IsNumber(2+x);`	`f-ja gražinanti "true", jei reiškinys susiveda į vieną skaitinę reikšmę. Šiuo atveju "false"`
	`IsInteger(15/5);`	`f-ja gražinanti "true", jei reiškinys sveikas skaičius. Šiuo atveju "true"`
	`F(x):= Eval(x+x)`	`F(x):= 2*x`
	`Expand((1+x)^5);`	`Reiškinio skleidimas polinomu`
	`Apply("+",{2,3});`	`Pritaikyti operatorių eilei argumentų.`
	`Taylor(x,0,5) Sin(x);`	`Skaičiuoja Teiloro skleidimą eilute pagal funkciją.`
	`Factors(x^2-1);`	`Faktorizuoja polinomą`
	`DiagonalMatrix({a,b,c});`	`Sukuria diagonalę matricą pagal nurodytus elementus`
	`Integrate(x,a,b) x*Sin(x);`	`Integruoja kintamąjį x nuo a iki b`
	`Ir kitos`	

2.2.2 Trūkumai ir privalumai

Trūkumai:

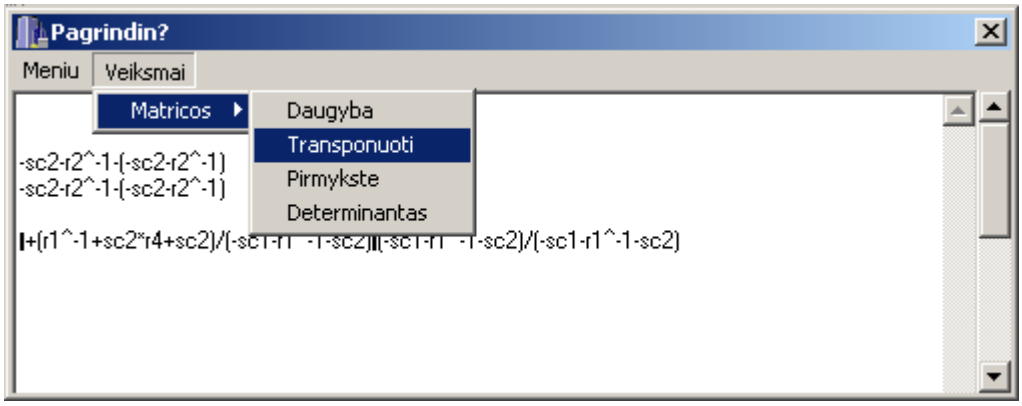
1) Funkcija gali išreikšti tik vieną nežinomąjį iš vienos tiesinės lygties.
2) Nesugeba apdoroti ilgų reiškinių.
3) Prastinant daromos klaidos.
4) Nėra galimybės išsaugoti ir pakrauti rezultatus.
5) Reikalauja griežtos sintaksės

Privalumai:

1) Turi papildomą f-ją reiškinių prastinimui.
2) Galimybė atsakymą pateikti "PrettyForm".
3) Galima per naudotis "Yacas" galimybėmis per WWW naršyklę.

2.3 "Simboliai" bibliotekos pritaikymas ir apžvalga

"Simboliai.dll" – biblioteka parašyta ir tobulinama Šiaulių universiteto magistranto M.Martinaičio. Biblioteka kol kas neturi jokių papildomų funkcijų. Atlieka reiškinių prastinimą ir bendravardiklinimą. Platinama kartu su "MS Windows" aplikacija(3.3.1. pav).



3.3.1. pav. “Simboliai.dll” MS Windows aplikacija

Pasak autoriaus, kuriant šį simbolinių skaičiavimų modulį buvo įgyvendinta galimybė pačiam vartotojui nuspręsti ar jis nori gauti subendravardiklintą atsakymą ar ne. Kadangi, kartais esant daug skirtingų simbolių, kurių neįmanoma suprastinti, bendravardiklinant atsakymas gaunamas didesnis už pateiktą sąlygą. Todėl suteikus vartotojui didesnę pasirinkimo laisvę galime gauti rezultatus priimtinesnius pačiam vartotojui.

2.3.1 Trūkumai ir privalumai

Trūkumai:

1) Nėra veiksmų tęstinumo.
2) Nėra galimybės išsaugoti ir pakrauti rezultatus.
3) Prastinant daromos klaidos.
4) Programa “lūžta” naudojant ilgesnius reiškinius.
5) Neergonomiška vartotojo sąsaja.
6) Prastinant reiškinius daromos klaidos

Privalumai:

7) Turi grafinę vartotojo sąsają(GUI) dirbančią MS Windows aplinkoje.

2.4 “Simbolinių” bibliotekų apibendrinimas

Visos anksčiau apžvelgtos bibliotekos yra daugiau ar mažiau skirtingos, juk jas kūrė visai skirtingi ir geografiškai tolimi žmonės. Tačiau jas riša bendras tikslas, prie kurio, kaip matysis iš atliktų tyrimų(žr.7.sk.),labiausiai priartėję Yacas programuotojai,- pažėrę visą galybę funkcijų(žr.2.2.1) yra platinama kartu su išeities kodu. Patikimiau prastina ilgas išraiškas, nors reiškinių prastinimas yra taipogi silpnoji programos vieta.

Interpreter – iš vokiečių kalbos išvertus reiškia,- interpretuotojas, aiškintojas. Turi kiek mažiau funkcijų ir galimybių nei Yacas, tačiau savo galimybėmis ir funkcijomis, kurias numato autorius sekančioje programos versijoje jei nepralenks, tai tikrai turėtų pasivyti Yacas. Skirtingai nei Yacas programą rašoma ir tobulinama vieno žmogaus. Tačiau pasak autoriaus, jis mielai sulauktų pagalbininkų.

Simboliai – tai jauniausia savo gyvavimu iš visų anksčiau peržvelgtų bibliotekų. Todėl nespėjusi dar priauginti papildomų funkcijų. Dabar platinama pirmoji “beta” programos versija, kur atliekamas reiškinių prastinimas ir bendravardiklinimas. Nors platinama su išeities kodu, tačiau programa skirta tik “MS Windows” operacinei sistemai.

3. REIKALAVIMŲ PROGRAMINEI ĮRANGAI APIBRĖŽIMAS

Problemos, kurias turi spręsti programinės įrangos inžinieriai yra labai sudėtingos, todėl prieš pradedant kurti sistemą yra surenkami reikalavimai sistemai ir atliekama jų analizė. Analizės galinis rezultatas yra reikalavimų specifikacija, kuri dažnai yra traktuojama kaip pirmas formalus dokumentas kuriant programinę įrangą. Reikėtų pabrėžti, kad vartotojo poreikiai ir reikalavimai nėra tas pats. Programuotojas, turėdamas tik vartotojo išreikštus poreikius bei pageidavimus, nesukurs geros ir priimtinos sistemos. Bet surinkus ir išanalizavus juos jau galima kalbėti apie problemos apibrėžimą.

Žinoma, kad geriau įveikti iškilusias problemas, vartotojas turi gauti programos prototipą. Reikalavimai skirstomi į dvi kategorijas:

1.Funkciniai sistemos reikalavimai.

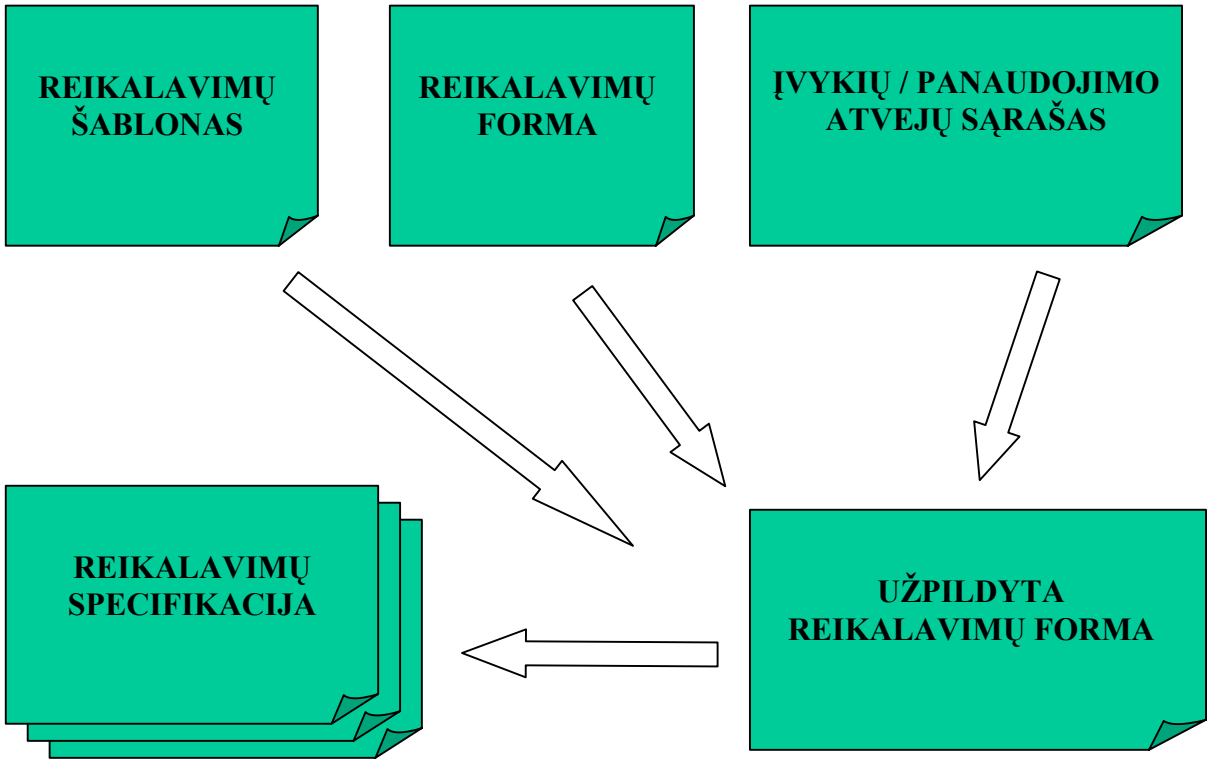
Jais aprašomas toks sistemos servisas, kokio tikisi sistemos vartotojas. Iš esmės vartotojui yra neįdomu kokiomis priemonėmis jis yra įgyvendinamas. Todėl šiame dokumente jie nebus aprašinėjami.

2.Ne funkciniai reikalavimai(4.1pav).

3.1 pav.*Nefunkcinių reikalavimų sąsaja su funkciniais reikalavimais*

Jie aprašo sunkumus, su kuriais susidūrusi sistema privalės veikti, ir standartus, kurie yra privalomi įdiegiant sistemą. Be to, naudojant natūralią kalbą, rašantysis ir skaitantysis ne visada vienodai interpretuoja tą pačią informaciją. Kartais, naudojant tiek natūralią kalbą, tiek lenteles bei diagramas, atsiranda painiava tarp bendrų idėjų ir detalių aprašymų, nes reikalavimų apibrėžimo parašymui naudojama informacija, turinti skirtingą detalumo lygį. Jei reikalavimų apibrėžimui naudojama natūrali kalba, iškyla keletas problemų: funkciniai reikalavimai, nefunkciniai reikalavimai ir projektavimo informacija nėra aiškiai atskiriami; kiekvienas paragrafas gali turėti keletą savų, individualių reikalavimų, kurie pateikti visai kitame pareiškime. Kartais praktikoje reikalavimų apibrėžimas yra sujungiamas su reikalavimų specifikacija. Laikas, reikalingas išanalizuoti reikalavimus ir sukurti didelę sistemą, gali būti net keleri metai. Per tokį laiko tarpą reikalavimai gali pasikeisti, o tai turėtų atsispindėti reikalavimų apibrėžime. Kad nereikėtų visko perrašinėti iš naujo, labai patogu yra vesti elektroninę dokumentaciją, juolab kad šiais laikais vien tik aparatūros pokyčiai yra milžiniški[18].

3.1 Reikalavimų specifikacijos sudarymas



•**Nefunkciniai reikalavimai** - nusako sistemos savybes, kuriomis ji turi pasižymėti. Tai kokybinės funkciniuose reikalavimuose numatytų funkcijų vykdymo charakteristikos. Jos dažnai atspindi vartotojo patirtį atliekant tam tikrą veiklą. Šie reikalavimai gali “išryškėti” bet kuriuo metu.

Tačiau patogiausia juos susieti su panaudojimo atvejais arba funkciniais reikalavimais, nustatant, kokiomis savybėmis ar kokybinėmis charakteristikomis šis funkcionalumo aspektas turi pasižymėti.

Nefunkciniams reikalavimams išskirti galima pasinaudoti toliau pateikiama proceso specifikacija.

3.1.1 Užsakovai ir kiti, sistema suinteresuoti asmenys

Užsakovas:

Docentas Gintautas Daunys

ŠU, Technologijos fakultetas, Elektronikos katedra

Elektronikos katedros vedėjas.

E-pašto adresas: g.daunys@tf.su.lt

Interneto svetainės adresas: http://silicis.su.lt/

Adresas: Vilniaus 141, Šiauliai.

Kiti suinteresuoti asmenys:

Uwe Dolinsky

Hochschule Wismar

Inžinierius

E-pašto adresas: u.dolinsky@iname.com

Interneto svetainės adresas: http://www.mb.hs-wismar.de/~dolinsky/casE.html

Adresas: Phillip Müller Straße Postfach 1210 23952 Wismar

3.1.2 Bendradarbiaujančios sistemos

Visa sistema dirbs šių tarpusavyje bendradarbiaujančių sistemų aplinkoje:

- Serverio pusėje: „Linux“ (operacinė sistema), „PostgreSQL“ (duomenų bazė);
- Kliento pusėje: „Windows“ aplinkoje su kitomis programomis gali sąveikauti per laikinąją atmintį („clipboard“).

3.2 Nefunkcinių reikalavimų specifikacija

Reikalavimai sudaromi pagal programinės inžinerijos Volere šabloną[18].

3.2.1 Reikalavimai sistemos išvaizdai

3.2.1.1 lentelė

Reikalavimas #: 1	Reikalavimo tipas: 1	Įvykis/panaudojimo atvejis #:
Aprašymas:	Automatizuota terpė turi turėti lengvą sąsają.	

Pagrindimas:	Parastas (nesudėtingas) panaudojimas.		
Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Prieinamumas, kad vartotojas nesivaržytų naudojantis sistema.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	4
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.2 Reikalavimai tinkamumui

3.2.2.1 lentelė

Reikalavimas #: 2	Reikalavimo tipas: 2	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Automatizuota terpe turi būti paprasta naudotis.		
Pagrindimas:	Paprastai panaudojimas, bet kokio asmens be apsimokymo (90% sėkmingas tinkamumas pirmu bandymu).		
Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Tyrimų veiklos našumo prieaugis, įprasti žymėjimai, nesudėtingi terminai, patogi navigacija, ŽB auto-gidas.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	4
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.3 Reikalavimai vykdymo charakteristikoms

3.2.3.1 lentelė

Reikalavimas #: 3	Reikalavimo tipas: 3	Įvykis/panaudojimo atvejis #:	
Aprašymas:	ŽB sistema turi palaikyti daugybę vartotojų, todėl transakcijų vykdymo laikas turi būti minimalus.		
Pagrindimas:	Automatizuota terpė su ŽB sistema turi greitai dirbi, nepriklausomai nuo prisijungusių vartotojų skaičiaus.		
Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Transakcijų greitis.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.3.2 lentelė

Reikalavimas #: 4	Reikalavimo tipas: 3	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Sistema duomenų mainams turi naudoti XML formatą.		
Pagrindimas:	XML duomenų formatas praplės sistemos gyvavimo laiką ir palengvins naujų sistemos dalių kūrimą.		
Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Stabilus duomenų mainų formatas.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.4 Reikalavimai veikimo sąlygoms

3.2.4.1 lentelė

Reikalavimas #: 5	Reikalavimo tipas: 4	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Automatizuotos sistemos klientai turi dirbti su kompiuteriais atitinkančiais sistemos reikalavimus.		
Pagrindimas:	Kad vartotojai galėtų naudotis maloniai, be stabdymų, tyrimų sistema, jie turi turėti atitinkamus kompiuterius.		
Šaltinis:	Vartotojas		
Tinkamumo kriterijus:	Kompiuteris atitinkantis sistemos minimalius reikalavimus (Pentium 150Mhz, 32MB ram).		
Užsakovo patenkinimas:	4	Užsakovo nepatenkinimas:	4
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.4.2 lentelė

Reikalavimas #: 6	Reikalavimo tipas: 4	Įvykis/panaudojimo atvejis #:	
Aprašymas:	ŽB serveris turi būti aukštos klasės kompiuteris.		
Pagrindimas:	Kad vartotojai galėtų naudotis maloniai, be stabdymų, reikalingas greitas serveris.		
Šaltinis:	Administratorius, vartotojas.		
Tinkamumo kriterijus:	Kompiuteris atitinkantis NP sistemos serverio minimalius reikalavimus (CPU 1500Mhz, 512MB ram, RAID).		
Užsakovo patenkinimas:	4	Užsakovo nepatenkinimas:	4

Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.5 Reikalavimai sistemos priežiūrai

3.2.5.1 lentelė

Reikalavimas #: 7	Reikalavimo tipas: 5	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Palaikymo darbai.		
Pagrindimas:	Aptiktų sistemoje klaidų taisymas, turi būti paliktas laiko rezervas nenumatytiems darbams.		
Šaltinis:	Užsakovas.		
Tinkamumo kriterijus:	Sutartis sistemos priežiūrai.		
Užsakovo patenkinimas:	4	Užsakovo nepatenkinimas:	4
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.6 Reikalavimai saugumui

3.2.6.1 lentelė

Reikalavimas #: 8	Reikalavimo tipas: 6	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Tik sistemos administratorius gali matyti konfidencialius duomenis.		
Pagrindimas:	Paprasti vartotojai negali matyti konfidencialių duomenų.		
Šaltinis:	Užsakovas.		
Tinkamumo kriterijus:	Konfidencialių duomenų apsauga.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.6.2 lentelė

Reikalavimas #: 9	Reikalavimo tipas: 6	Įvykis/panaudojimo atvejis #:
Aprašymas:	Vartotojas turi gauti informaciją pagal savo statusą.	
Pagrindimas:	Vartotojas, besinaudojantis sistema, turi jaustis saugus dėl asmeninių duomenų.	
Šaltinis:	Užsakovas.	
Tinkamumo kriterijus:	Sistemoje laikomi asmeniniai duomenys prieinami tik pačiam asmeniui ir sistemos	

	administratoriui.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.6.3 lentelė

Reikalavimas #: 10	Reikalavimo tipas: 6	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Sistema turi mokėti daryti rezervines kopijas.		
Pagrindimas:	Sistema turi būti atspari nenumatytiems trukdžiams.		
Šaltinis:	Užsakovas.		
Tinkamumo kriterijus:	Sistemoje laikomi duomenys archyvuojami, daromos rezervinės kopijos.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.7 Kultūriniai-politiniai reikalavimai

3.2.7.1 lentelė

Reikalavimas #: 11	Reikalavimo tipas: 7	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Sistema turi palaikyti daugiakalbystę. Visi vartotojo interfeisai turi būti lengvai pritaikomi įvairioms kalboms.		
Pagrindimas:	Sistema gali būti naudojama ne tik Lietuvoje.		
Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Galima keisti sistemos sąsajos kalbą.		
Užsakovo patenkinimas:	3	Užsakovo nepatenkinimas:	3
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.7.2 lentelė

Reikalavimas #: 12	Reikalavimo tipas: 7	Įvykis/panaudojimo atvejis #:
Aprašymas:	Sistemoje negalima naudoti ne korektiškų terminų ar išsireiškimų.	
Pagrindimas:	Naudojant kokius nors ne korektiškus terminus ar išsireiškimus, galima įžeisti vartotoją.	

Šaltinis:	Vartotojas.		
Tinkamumo kriterijus:	Ne korektiški terminai, išsireiškimai.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

3.2.8 Teisiniai reikalavimai

3.2.8.1 lentelė

Reikalavimas #: 13	Reikalavimo tipas: 8	Įvykis/panaudojimo atvejis #:	
Aprašymas:	Sistemos autorinės teisės priklauso “Šiaulių universitetui”		
Pagrindimas:	Užsakovas nori, kad produktas būtų laisvai platinimas.		
Šaltinis:	Užsakovas.		
Tinkamumo kriterijus:	Licenzija.		
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:	Užregistruotas 2004 m. gegužės 25 d.		

4. VARTOTOJO SĄSAJOS FORMAVIMAS

4.1 Sąsajų apžvalga

Žmogaus ir kompiuterio sąveikoje didelį vaidmenį vaidina vartotojo sąsaja, kurią renkantis reikia pastebėti, kad grafinė sąsaja yra daug sudėtingesnė. Todėl žymi resursų dalis turi būti panaudota kuriant tokią sąsają.

Šiomis dienomis dauguma programinės įrangos kūrėjų siūlo savo sąsajų valdymo sistemas, kurios yra nesuderinamos tarpusavyje. Tai reiškia, kad daug pastangų sunaudojama perkeliant aplinką nuo vienos mašinos prie kitos.

Dabar jau pradedama standartizacija ir kaip standartas naudojamas X-window sąsaja. Šis standartas yra palaikomas visų darbo stočių, bet tai dar nėra personalinio kompiuterio standartas[19].

4.1.1 Grafinės sąsajos apžvalga

Vartotojo sąsajos dažniausiai naudoja langus, krintančius meniu, piktogramas, nuorodas tai yra kasdieniškas dalykas personaliniame kompiuteryje ir darbo stotyje.

Terminas 'WIMP sąsaja' (paveldėtas iš langų, piktogramų, krintančių meniu ir nuorodų) buvo naudojama kaip standartinė šios sąsajos tipui. Dabar ji vadinama grafinė vartotojo aplinka (GUIs). Ji charakterizuojama taip:

1.Skirtingi langai, esantys vartotojo ekrane, leidžia vienalaikiškai atvaizduoti skirtingą informaciją.

2.Piktograminis informacijos vaizdavimas.Vienose sistemose piktogramos vaizduoja laikmenas, kitose- procesus.

3.Komandų išrinkimas per meniu.

4.Rodantysis įrenginys, pavyzdžiui pelė, naudojamas pasirenkant punktus iš meniu arba pažymėti kažkokį tai punktą lange.

5.Galimybė atvaizduoti tiek tekstinę, tiek grafinę informaciją.

Grafinės sąsajos pranašumas:

1.Lengva išmokti naudotis. Vartotojas neturintis darbo su kompiuteriu patyrimo gali išmokti naudotis po trumpo mokymo kurso.

2.Vartotojas gali turėti sistemos sąsajos keletą langų. Persijungiant nuo vieno lango prie kito neprarandama informacija esanti pirmame lange.

Grafinė aplinka yra daug sudėtingesnė. Žymi resursų dalis turi būti panaudota kuriant tokią sąsają.

Šiomis dienomis dauguma programinės įrangos kūrėjų siūlo savo sąsajos valdymo sistemas, kurios yra nesuderinamos tarpusavyje. Tai reiškia, kad daug pastangų sunaudojama perkeliant aplinka nuo vienos mašinos prie kitos. Dabar jau pradedama sąsajų standartizacija ir kaip standartas naudojamas X-window aplinka. Šis standartas yra palaikomas visų darbo stočių, bet tai dar nėra personalinio kompiuterio standartas. Grafinės vartotojo sąsajos priemonės darbo stotims dažniausiai pagrįstos X-windows[19].

4.1.2 Tiesioginio manipuliavimo sąsajos apžvalga

Tiesioginio manipuliavimo interfeisas siūlo vartotojui modelį informacinių laukų, kuriuos vartotojas gali tiesiogiai keisti. Tokiame modelyje vartotojui nebūtina aiškinti informacijos apdorojimo komandų.

Paprasčiausias tiesioginio manipuliavimo pavyzdys, tai teksto redaktorius. Norint įterpti tekstą, kursorius perkeliamas į norimą vietą ir įvedamas tekstas. Vesdamas tekstą vatotojas gali iškarto matyti rezultatą.

Tiesioginio manipuliavimo pranašumai:

1. Vartotojas jaučia, kad gali kontroliuoti kompiuterį ir nesibaimina jo.
2. Paprastai tokiam interfeisui vartotojo apmokymas yra paprastas.
3. Vartotojas betarpiškai grįžta prie savo veiksmų, todėl klaidos gali būti aptiktos ir ištaisytos labai greitai.

Tiesioginio manipuliavimo sąsaja reikalauja vartotojo informacinių laukų modelio. Vartotojas redaguoja modelį, kurį betarpiškai taiso, pabraukia tam tikrą informaciją.

Problema susijusi su tiesioginio manipuliavimo sąsajos kūrimu, kai vartotojai turi didelį informacinį lauką, ir jie gali judėti po visą šį lauką ir visada būti informuotiems apie einamą poziciją?

Kompiuterines sistemas vis labiau naudoja atsitiktiniai ir neapmokyti vartotojai, todėl tiesioginio manipuliavimo sąsaja vis daugiau naudojama[19].

4.1.3 Komandinės sąsajos apžvalga

Komandinė sąsaja reikalauja vartotojo įvesti tekstinę komandą sistemai. Komanda gali būti klausimo pavidalo, kokios nors komandų posistemės įvedimas arba ji gali iškviesti kitų komandų seką.

Komandinės sąsajos privalumai yra:

1. Ji gali būti realizuojama ir pigiuose, tekstiniuose monitoriuose.
2. Kalbos analizavimo metodai yra gerai išvystyti dėl darbo nuveikto kuriant kompiliatorius. Sukurti komandinės kalbos apdorojimą yra palyginti nesunku.
3. Beveik bet kokio sudėtingumo komandos gali būti sukurtos sujungiant savas komandas. Galinga UNIX operacinės sistemos ypatybė yra galėjimas joje rašyti komandinės kalbos programas (shell programs).
4. Sąsaja gali būti iš trumpų komandų, kad vartotojui reikėtų trumpiau surinkinėti komandas.

Trūkumai :

1. Vartotojai turi išmokti komandinę kalbą, kuri kartais yra labai sudėtinga. Kai kuriais atvejais (kaip UNIX shell kalba) tik keletas vartotojų pilnai išmoksta kalbą.
2. Vartotojai surinkdami komandas neišvengiamai daro klaidų. Tai reikalauja klaidų apdorojimo ir pranešimų generavimo galimybės įtraukimo į komandinę kalbą. Meniu sistemoje vartotojai negali įvesti neteisingo veiksmo, taigi sistema tuo atveju visada pripažįsta, kad įvedimas yra teisingas.
3. Sąsaja su sistema bendrauja naudojant klaviatūrą. Sąsaja negali naudotis nurodomaisiais įrenginiais kaip pvz. pelė.

Nerūpestingiems ir nepatyrusiems vartotojams, komandinė sąsaja yra netinkama. Laikas, reikalaujantis išmokti komandų kalbą, yra neproporcingas laikui, praleistam dirbant kompiuteriu. Tokiems vartotojams meniu pagrįsta sąsaja (arba kartais natūralios kalbos sąsaja) yra vienintelė priimtina rūšis.

Patyrę, pastovūs kompiuterių naudotojai kartais teikia pirmenybę komandomis pagrįstoms sąsajoms. Komandų sąsaja leidžia greitesnę sąveiką su kompiuteriu ir supaprastina sudėtingų komandų įvedimą.

Patyrę vartotojai gali net pageidauti komandų apjungimo į procedūras ir programas.

Projektuojant komandinės kalbos sąsają, turi būti priimama daug projektavimo sprendimų:

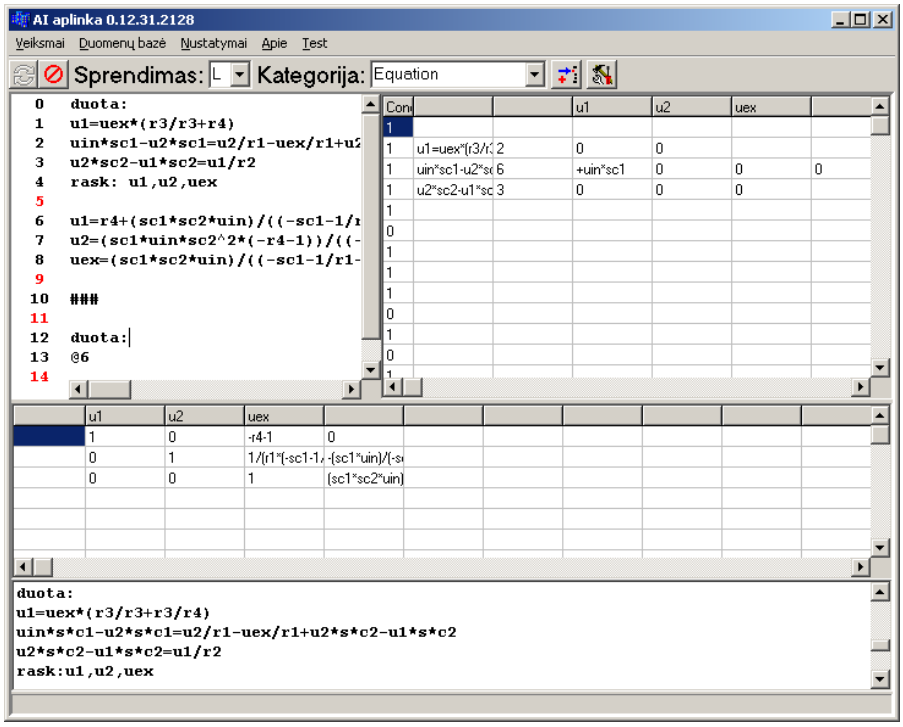
1. Ar bus įmanoma apjungti komandas ir sukurti naujas komandas-procedūras? Tai galinga galimybė patyrusio vartotojo rankose, bet tikriausiai nereikalinga daugumai sistemos vartotojų.
2. Kokia mnemonika turi būti pasirinkta sisteminėms komandoms? Projektuotojas turi išbandyti ir sugalvoti prasmingus komandų pavadinimus, išlaikant juos trumpais, kad sumažinti surinkinėjimo laiką.

3. Ar vartotojams bus suteikta galimybė pervadinti komandų vardus, kurios atitiktų jų pačių poreikius?

Pervadinimo privalumas tas, kad komandos vardas gali būti trumpas patyrusiems vartotojams arba išplėstas tiems, kurie ką tik susipažinę su sistema. Tačiau 'pc' ir 'Compile-with-Pascal' gali atitikti tą pačią operaciją, būtent Paskalio kompiliatoriaus inicializavimą. Tokio plataus komandų pervadinimo trūkumas- vartotojai nebenaudoja bendros kalbos bendravimui su sistema. Žinoma, komandinė ir meniu pagrįsta sąsaja nėra abipusiškai puiki. Daug didelių programinės įrangos sistemų turi įtikti didelei vartotojų grupei: nuo patyrusių kompiuterių profesionalų iki naudotojų be jokio kompiuterinio programos pagrindo. Šiaip įmanoma surišti kelių skirtingų vartotojo sąsajų apdorojimus taip, kad skirtingas įvedimo tipas gali būti paruoštas skirtingoms vartotojų grupėms[19].

4.1.4 Sukurtos sąsajos apžvalga

Atsižvelgiant į anksčiau apžvelgtų sąsajų privalumus ir trūkumus realizuota teorinių tyrimų terpės sąsaja(4.1.4.1 pav.) visiškai patenkino siekiamo naujo ir plačiai suderinamo modelio lūkesčius. Vartotojo apmokymas naujai sąsajai paprastas. Vartotojas gali betarpiškai grįžti prie savo veiksmų. Ją sudaro trumpų komandų rinkinys, atsižvelgiant į teksto surinkimo laiką rašant komandas.



4.1.4.1 pav. Automatizuotos tyrimų terpės vartotojo sąsaja

Detalus sukurtosios automatizuotos teorinių tyrimų terpės aprašymas pateiktas pirmame priede(Priedas 1).

4.2 Klaidų pranešimų projektavimas

Pirmas vartotojo įspūdis apie kompiuterinę sistemą dažniausia yra sistemos išvedami pranešimai apie klaidą.

Nepatyrę vartotojai pradėję dirbti ir padarę pirmą klaidą gali ne iš karto suprasti pasirodžiusio pranešimo apie klaidą. Toks nesupratimas gana sunkiai įsivaizduojamas įgudusiems programinės įrangos kūrėjams, bet tai tikrai dažnai įmanoma nepatyrusiems ar atsitiktiniams sistemos vartotojams.

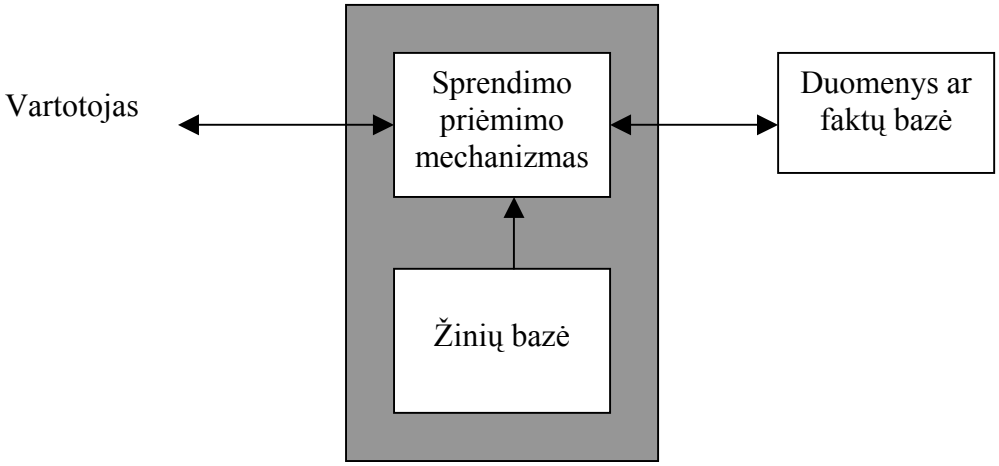
Pranešimai apie klaidą turi būti mandagūs, glausti, nuoseklūs ir konstruktyvūs. Jie privalo būti neįžeidžiantys ir neturėti garsinio pavidalo, kuris gali sutrikdyti vartotoją. Kai tik įmanoma, pranešimas turi bandyti nuspėti klaidą, paaiškinti kaip ji galėtų būti ištaisyta. Jei tai tinkamas būdas, vartotojas turi turėti galimybę išsikviesti kontekstinę pagalbą, kuri išvestų daugiau informacijos apie padarytą klaidą.

Automatizuotos teorinių tyrimų terpės generuojami klaidų pranešimai su paaiškinimais pateikti pirmame priede (3.1.1, 3.1.2 ir 3.1.3 lentelėse).

5. ŽINIŲ BAZĖS TEORINIS MODELIS

5.1 Bendros žinios apie žinių bazes

Susidūrę su kliūtimi, žmonės, bandydami ją išspręsti, naudoja įvairias žinias. Programos, paremtos žinių bazių sistemomis, sudarytos iš *sprendimų priėmimo mechanizmo* ir *Žinių bazės.* Šiose programose glaudžiai susijusios *duomenų* ir *Žinių bazės* (pav. 6.1). Sprendimų priėmimo mechanizmas manipuliuoja žinių bazėje esančiais duomenimis ir faktais, turėdamas tikslą išspręsti problemą, aprašytą duomenų bazėje esančia informacija. Sprendimų priėmimo mechanizmas turi bendras žinias problemų sprendimui (t.y. kaip pasiekti problemos sprendimą) kai žinių bazėje yra žinios apie šią individualią problemą.



5.1. pav. *Bendra žinių bazių sistemos struktūra*

Žinių bazėmis pagrįstų sistemų galimybės pradėjo vystytis atskyrus žinių bazę nuo jos panaudojimo. Šis atskyrimas leidžia suprojektuoti naujus pritaikymus sukūrus tik naują žinių bazę kiekvienam atvejui. Sprendimų priėmimo mechanizmas paliekamas tas pats. Vieną kartą užkoduotas sprendimo mechanizmas gali būti taikomas kitoje srityje, tuo smarkiai supaprastinant projektavimo procesą.

Žinios gali būti įvairių formų. Vienos žinios susideda tik iš faktų, kitos nustato ryšius tarp faktų. Vienos žinios yra algoritminės, kitos - *euristinės* (euristika - apytikslis skaičiavimas, strategija ar metodai, naudojami pagerinti efektyvumą sistemos, kuri bando rasti sudėtingos problemos sprendimą). Nesvarbu kokios formos žinios, kad būtų galima apdoroti jas kompiuteriu, reikia atvaizduoti jas atmintyje ir sukurti priemones jų interpretavimui.

Žinių bazėse paprastai naudojamos penkios pagrindinės žinių atvaizdavimo schemos: logika, taisyklės, asociatyviniai (semantiniai) tinklai, freimai ir objektai. Kiekviena schema tinkama skirtingiems uždavinių tipams ir panaudoja skirtingus būdus žinių interpretavimui[20].

5.2. Paieška - intelektikos pagrindas

Pirmųjų intelektikos (AI - *artificial intelligence*) tyrinėtojų tikslas buvo rasti metodus sprendimui tokių uždavinių, kai neparanku taikyti esančias kompiuterines technologijas, nors lėtesnės žmogaus smegenys atsakymą randa gana lengvai. Kaip taisyklė, šie uždaviniai arba neturi žinomo algoritminio sprendimo arba yra jie tokie sudėtingi, kad nepraktiška įgyvendinti juos kompiuteryje. Vis dėlto, šiuos uždavinius dažnai spendžia žmonės su "žemesnio lygio" duomenų apdorojimo galimybėmis. Taigi, reikia sukurti naują sprendimų paieškos metodą, panašų į tą, kurį naudoja žmogus ir įgyvendinti jį kompiuteryje. Vienas iš svarbiausių metodų yra *paieška*.

Paieška duomenų struktūrose visada buvo fundamentali kompiuterinių algoritmų koncepcija. Tačiau, AI projektuotojų kuriama paieška yra kitokia, ji skirta paieškai *probleminėje srityje* ne kokios nors duomenų dalies, bet kelio, jungiančio pradinį sąlygos aprašymą su reikiama atsakymo būsena (t.y. su išspręstu uždaviniu). Šis kelias atspindi sprendimo žingsnius. Sprendimo paieškos procese randama *sprendinių aibė* (t.y. dalis probleminės srities, kuri buvo ištirta).

Skirtingai nuo duomenų struktūrų, kurios yra apibrėžtos ir jau egzistuoja paieškai prasidėjus, probleminė sritis dažniausiai, nors ir ne visada, yra *procedūriškai apibrėžta*. Tai reiškia, probleminė sritis nėra sukuriama po to atliekant paiešką, bet sukuriama jos tyrinėjimo bėgyje. Procedūros naudojamos tikslu nustatyti kitas galimas srities būsenas, į kurias galima pereiti iš esamos būsenos, ir tik ištyrinėti keliai turi būti tiksliai apibrėžti.

Probleminė sritis gali būti pavaizduota kaip *tinklas* arba *medis*. Tinklas (grafas) yra netuščia viršūnių aibė, lankų aibė ir lankų išsidėstymas tarp viršūnių porų. Kiekviena viršūnė yra problemos būsena. Lankai aprašo veiksmus, kurie gali būti atlikti pereinant nuo vienos būsenos prie kitos. Perėjimas per tinklą atliekamas judant tarp viršūnių tikintis atsirasti vis arčiau tikslo. Medis yra kryptingas, beciklis tinklas, kuriame kiekviena viršūnė turi vieną tėvą. Medžio šaknis yra pradinė problema ir kiekviena šaka atitinka galimą sprendimą[20].

5.3 Žiniomis pagrįstos sistemos

Bendrais žodžiais, žiniomis pagrįstos sistemos gali būti apibrėžta kaip:

Kompiuterizuota sistema, naudojanti žinias apie tam tikrą sritį su tikslu gauti uždavinio iš tos srities sprendimą. Iš esmės šis sprendimas yra tas, kurį suranda toje srityje nusimanantis žmogus, sprendžiantis tokį pat uždavinį.

Nors žiniomis pagrįstų sistemų apibrėžimas yra pateiktas griežtai, daug įprastų programinės įrangos sistemų gali būti neteisingai priskirtos prie žiniomis paremtų sistemų. Pavyzdžiui, panagrinėkime įprasta programavimo kalba (pvz.: FORTRAN, PASCAL) parašytas programas, kurios skaičiuoja sroves ir įtampas elektrinėse schemose, analizuoja įtempimo faktorius tiltų lynams, skaičiuoja paskolų riziką, ar imituoja transporto srautus mieste siekiant nustatyti optimaliausią šviesoforų konfigūraciją. Visos šios programos atlieka tokius pat veiksmus kaip ir ekspertas (veikia daug greičiau!) ir naudoja tas pačias žinias (t.y. formules)[21].

Tokios programos nepriklauso žiniomis pagrįstomis sistemomis. Priežastis yra ta, kad tos sistemos nėra paprastas eksperto žinių dubliavimas specifinėse srityje. Reikalingas geresnis apibrėžimas šiam neatitikimui išspręsti. Išskirkime tris koncepcijas tam, kad detaliau aprašyti, kas yra žiniomis pagrįsta sistema.

Trys fundamentalios koncepcijos išskiria žiniomis pagrįstas sistemas nuo kitų įprastinių algoritminių ir paieška pagrįstų programų:

1. Žinių atskyrimas nuo jų panaudojimo

2. Ypač specifinės sferos žinių panaudojimas

3. Dažniau euristinė, nei algoritminė manipuliacija duomenimis

5.4. ŽB sistemų savybės

Privalumai (+):

1. Universalumas. ŽB sistemos leidžia plačiai panaudoti ekspertų įgytas žinias ir įgūdžius už mažesnę kainą. Pavyzdžiui, firmoje įvairioms grupėms gali reikti žinių apie mokesčių įstatymus, o yra tik vienas tos srities ekspertas. Tada galima kurti mokesčių žinių sistemą. Ja galės naudotis daugelis žmonių ir tai mažiau kainuos nei samdyti dar vieną ekspertą.

2. Lengvai modifikuojamos. Žinių atskyrimas nuo sprendimus priimančio mechanizmo supaprastina jų modifikavimą. Tai aktualu, nes žinios dažnai kinta.

3. Sprendinio patikimumas. ŽB sistemos visada pateikia vienodus sprendimus, o eksperto sprendimas gali priklausyti nuo jo emocinės ar sveikatos būsenos.

4. Nepertraukiamas darbas. Gali dirbti 24h per parą , taip pat savaitgaliais.

5. Išsaugomos žinios. Žinios išlieka netekus eksperto, o atėję nauji ekspertai gali jomis pasinaudoti.

6. Galima priimti sprendimus, esant nepilnai informacijai.

7. Sprendimai yra paaiškinami.

Žinių sistemų trūkumai (-):

1. Sprendimai ne visada teisingi. Ekspertai dažnai daro klaidas, todėl tikėtina, kad žinių sistemos taip pat daro klaidas.

2. Problemiškai orientuotos. Žinių sistemos turi ribotas žinias, bet jos vis tiek stengiasi rasti sprendimą, nors ir klaidingą. O žmogus žino savo žinių ribas ir nebandys išspręsti problemos, jei apie ją neturi informacijos.

4. Trūksta bendro supratimo apie probleminę sritį.

6. ŽINIŲ BAZĖS PRAKTINIS MODELIS

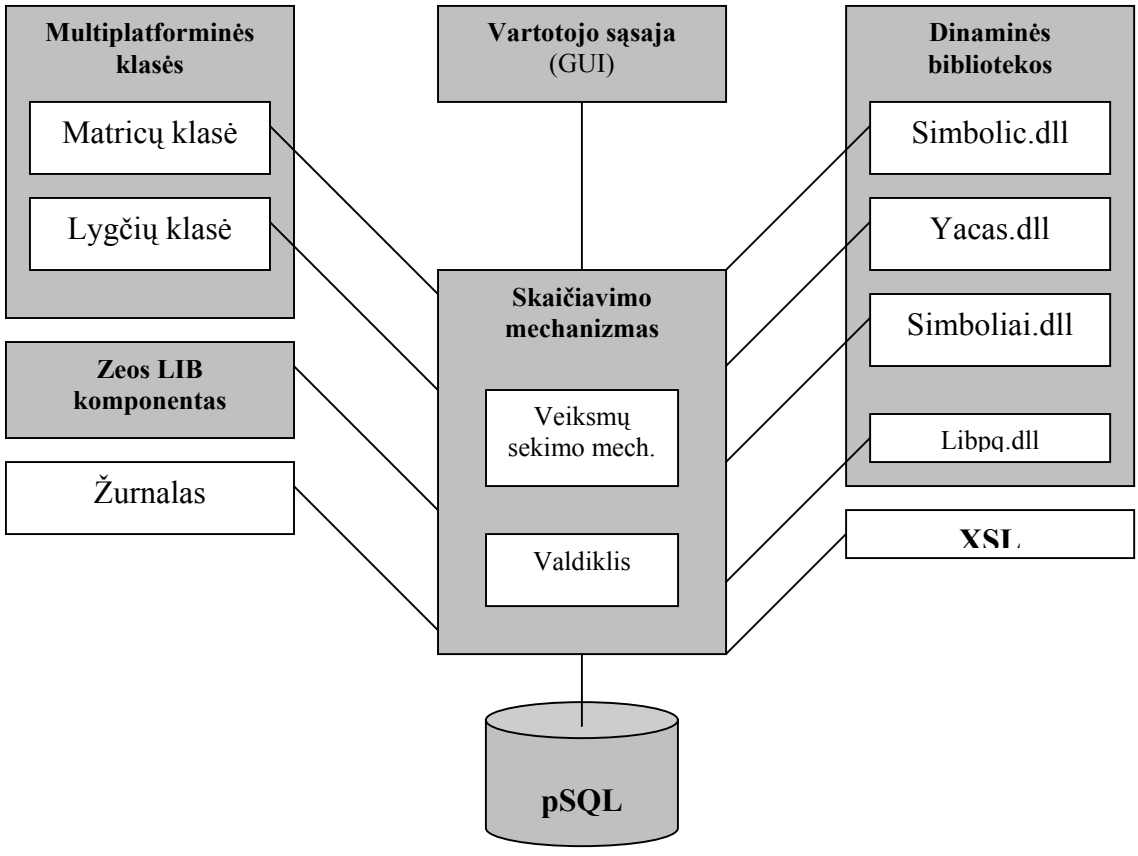
Atliekant įvairiausius tyrimus svarbiausia yra nustatyti pagrindinį uždavinio formulavimo kriterijų, pagal kurį jis būtų tinkamas apdoroti žinių bazių sistemomis. Uždavinys, atitinkantis visus šiuos kriterijus, yra laikomas tinkamu tokiam apdorojimui. Pasirodo kartais yra lengviau problemos sprendinį gauti, pasinaudojus tradiciniais programavimo metodais, o ne žinių bazėmis. Žinių bazių galimybių įvertinimo kriterijai turi būti išnagrinėti prieš pradedant spręsti uždavinį. Šie kriterijai gali būti grupuojami į dvi pagrindines kategorijas:

1) Formuluotės atitikimas
2) Galimybė gauti visus reikalaujamus resursus.

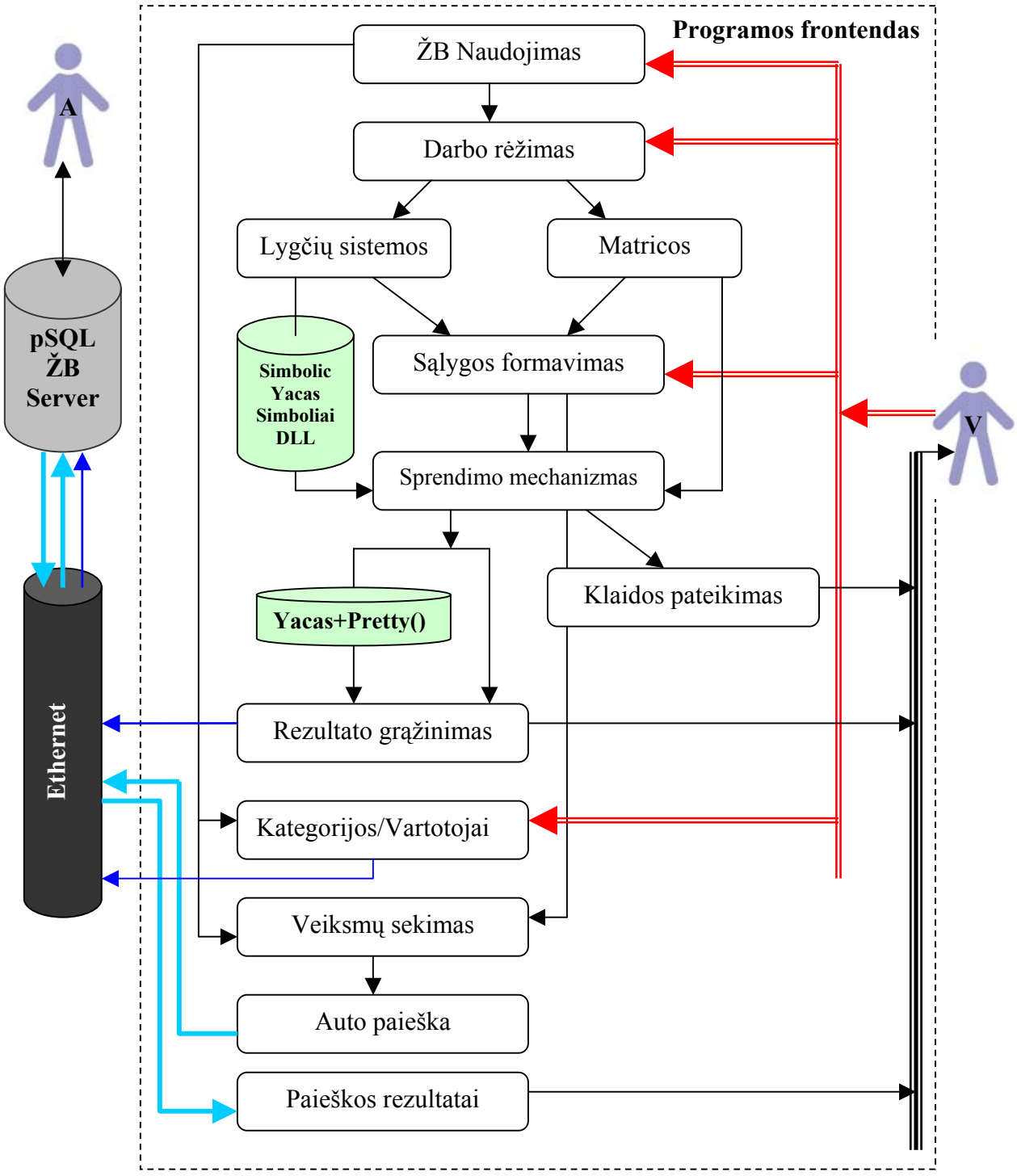
Formuluotė yra viena pagrindinių uždavinio formulavimo kriterijų, kurios tinkamumą užtikrina:

1) Taisyklinga sintaksė
2) Teisinga sąlyga

Apie formuluotę plačiau(ž.r. priedas 1). Resursų gavimo galimybę užtikrina sistemos reikalavimai veikimo sąlygoms (ž.r 3.2.4.1 lentelė).



6.1 pav. Supaprastinta automatizuotos teorinių tyrimų terpės programinio modelio shema



6.2 pav. Supaprastinta duomenų srautų judėjimo diagrama

6.1.1 lentelė

Paaiškinimai

	Duomenų srautas tarp vidinių procedūrų
	Įeinanti informacija
	I/O SQL duomenų srautai
	Išvedama informacija
	Įėjimo SQL duomenų srautas

Supaprastinta automatizuotos teorinių tyrimų terpės programinio modelio shema pavaizduota 6.1 pav., kurią sudaro: *valdiklis*,- pagrindinis programinių srautu reguliavimo mechanizmas; *veiksmų sekimo mech.,*- pastoviai seka klavišų paspaudimus; *skaičiavimo mech.* – skaičiavimo algoritmų rinkinys. Skaičiavimo mechanizmas bendrauja su multiplatforminėmis klasėmis ir dinaminėmis biliotekomis. Bet kokio skaičiavimo uždavinio sprendimo eigą galima rasti žurnale(LOG). Gauti sprendimų rezultatai talpinami į žinių bazių serverį ir atrenkami panašūs sprendimai naudojant "ZeosLIB" komponentą bei "libpq.dll dinaminę biblioteką. Supaprastinta duomenų srautų judėjimo shema pateikta 6.2 pav. o paaiškinimai 6.1.1 lentelėje.

6.1 PostgreSQL

PostgreSQL(pSQL) – objektiškai orientuotų duomenų bazių valdymo sistema, kuri buvo pradėta plėtoti nuo 1986m. iki šių dienų, kai pSQL tapo profesionalia atviro kodo duomenų baze. PSQL kilo iš Ingres duomenų bazės, kuri buvo plėtojama Kalifornijos universitete(1977-1985m.). Nuo 1986 iki 1994 metų Maiklas Stronebeker(Michael Stronebraker) būrė komadą, kuri užsiėmė objektiškai orientuoto duomenų bazės serverio kūrimu. Jį pavadino Postgres (iš lotyniško žodžio *post*, reiškiančio po; PostgreSQL buvo išdirbta po Ingres) .

Pagrindinis pSQL plėtotojų tikslas buvo tobulinti pačia sistemą ir naujų funkcijų integraciją. Nors pSQL galima naudoti ir lokaliai, tačiau ši duomenų bazė orientuota į kliento/serverio darbo rėžimą[22].

6.1.1 Sistemos klientas-serveris panaudojimas

Klientas/Serveris yra palyginti naujas DB modelis, kuris daugiavartotojiškose duomenų bazėse atlieka darbo paskirstymą tarp keleto kompiuterių.

Kliento /serverio pagrindiniai komponentai:

- serveris,
- kliento priedai,
- programinė įranga, skirta palaikyti ryšį tarp kliento ir serverio.

Serveris valdo visą DB aptarnavimo mechanizmą . Jo funkcijos yra :

1.aptarnauti bent vieną duomenų bazę, su kuria vienu metu dirba keletas vartotojų;

2.reguliuoti priėjimą prie duomenų bazės,

3.organizuoti duomenų apsaugą, taikant įvairias apsaugos formas:

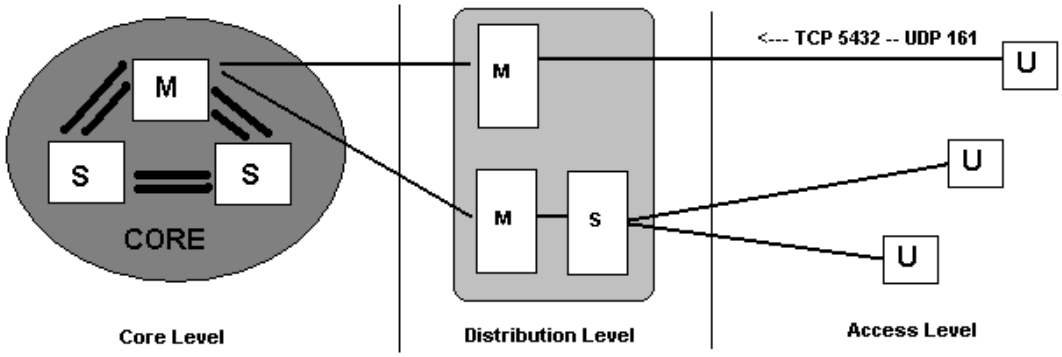
- SQL*Plus Product_User_Profle table;
- Roles ir privilegijas, o taip pat duomenų archyvavimo ir kopijų atstatymo metodus.

Klientas- sistemos dalis, kuri vykdo kliento priedus ir leidžia prisijungti prie DB, esančios serveryje. Kliento priedai gali būti vykdomi tame pačiame kompiuteryje kur yra serveris, bet taip pat kliento priedai gali būti vykdomi kliento kompiuteryje.

Kliento priedai-išorinis interfeisas(frontendas)- yra dalis sistemos, kurią vartotojas naudoja duomenų apdorojimui. Juose taip pat numatyta visa duomenų apdorojimo logika, duomenų teisingumo analizė. Klientas siunčia, gauna duomenis iš serverio bei juos analizuoja. Sistemoje klientas/serveris, klientas dirba ne su visu failu, o tik su atskirais jo elementais, kaip pavyzdžiui lentelės eilutėm (įrašais).

6.2 Žinių bazės modelio struktūra

Konkrečiu atveju žinių bazės modelį sudaro vienas ar keletas paraleliai dirbančių tarnybinių stočių, kurių duomenų bazės(PostgreSQL[23]) stekuotos nuo pagrindinės(Master) sudarydamos vieną bendrą(CORE Level). Taip pat viena ar keleta tarpinių stočių (Distribution Level) į kurias tiesiogiai kreipiasi vartotojų frontendinės aplikacijos(Access Level) pasirinkdamos joms artimesnę ir mažiau apkrautą stotį pagal ICMP(RFC792) ir SNMP(RFC1157)[24] protokolus. Uždavinys ir jo atsakymas talpinamas į žinių bazę, kur indeksuojamas pagal kategorijas. Kategorijas užsiduoda pats vartotojas. Naudojant „POSIX regular expression“[25] paieškos pagal indeksavimą kriterijus,- surandamas kuo artimesnis sprendimo variantas apie kurio radimą praneša žinių bazės asistentas. Rezultatas pakraunamas naudojant Distribution Level tarnybinių stočių VIEW`us ir funkcijas, kurie pagreitina paieškos ir rezultatų atidavimą bei vartotojui pageidaujant panašius tyrimų rezultatus jums gali pristatyti į elektroninį paštą. Tokia žinių bazė tarsi pati apsimoko be papildomų pastangų ar apmokymų kaupdama vartotojų siunčiamą informaciją apie atliekamus tyrimus.



6.2 pav. Bendra žinių bazių sistemos praktinio modelio struktūra

Kokie turėtų būti uždavinių formulavimo kriterijai, kad žinių bazių sistema juos galėtų tinkamai apdoroti?

1) Uždavinio sąlyga visada turi prasidėti žodeliu „duota:“, šis žodelis yra esminis formuluojant uždavinio sąlygą ir saugant informaciją XML, XSL ar TXT formatais.

Sąlygos teksto įvedimo kriterijus apibrėžia lygčių, jų sistemų ir simbolinių matricų šablonai bei keletas neesminių taisyklių apie kurių nepaisymą vartotojas gali būti informuojamas tiesiogiai.

2) Sprendimas gaunamas parašius žodžius „rask:“ ar „pateik:“ priklausomai nuo darbo aplinkos. Siekiant gerų sprendimo rezultatų, svarbu nurodyti kintamąjį ar jų seką, kuriuos norime rasti, bei raktinį žodį, pagal kurį bus indeksuojamas ar pateikiamas indeksaciją atitinkantis sprendinys, kaip pagalbinis rezultatas.

Bendras žinių bazių modelis pavaizduotas 6.2.pav.. Buvo ištirta, kad viena trancakcija su užklausa ir atsakymu(round-trip) maksimaliai sunaudoja 0,3kbps ryšio kanalo pralaidumo juostos. Tai reiškia, kad vidutiniškai galimų vienalaikių trancakcijų skaičius iš skirtingų IP būtų lygus:

F – ryšio kanalo pralaidumo juostos plotis kilobitais per sekundę(kbps)

z – trancakcijų skaičius

$$z = \frac{F}{0,3} - 2\% \qquad (6.2.1)$$

Iš 6.1.1 formulės seka, kad turint 256kbps ryšio kanalą galima vykdyti 836 vienalaikes trancakcijas iš skirtingų taškų. Ne paslaptis, kad kiekvienas IPT naudoja srauto “overbook”. Laikoma, kad “overbook” esant 1:30 ryšio kokybė dar nenukenčia. Tai reiškia, kad serveris turėdamas 256kbps ryšio kanalą teoriškai sugebėtų aptarnauti apie 25000 frontendų. Iš to seka, kad taikant “overbook” 1:30 $\frac{1}{30}$ vartotojų yra užėmę ryšio kanalą paros bėgyje.

Jei prisijungimų prie bazės skaičius neviršija 3000 vienalaikių tai „Core“ ir „Distribution“ lygius puikiai realizuos viena tarnybinė stotis su įdiegtu QoS(Quality Of Service) palaikymu.

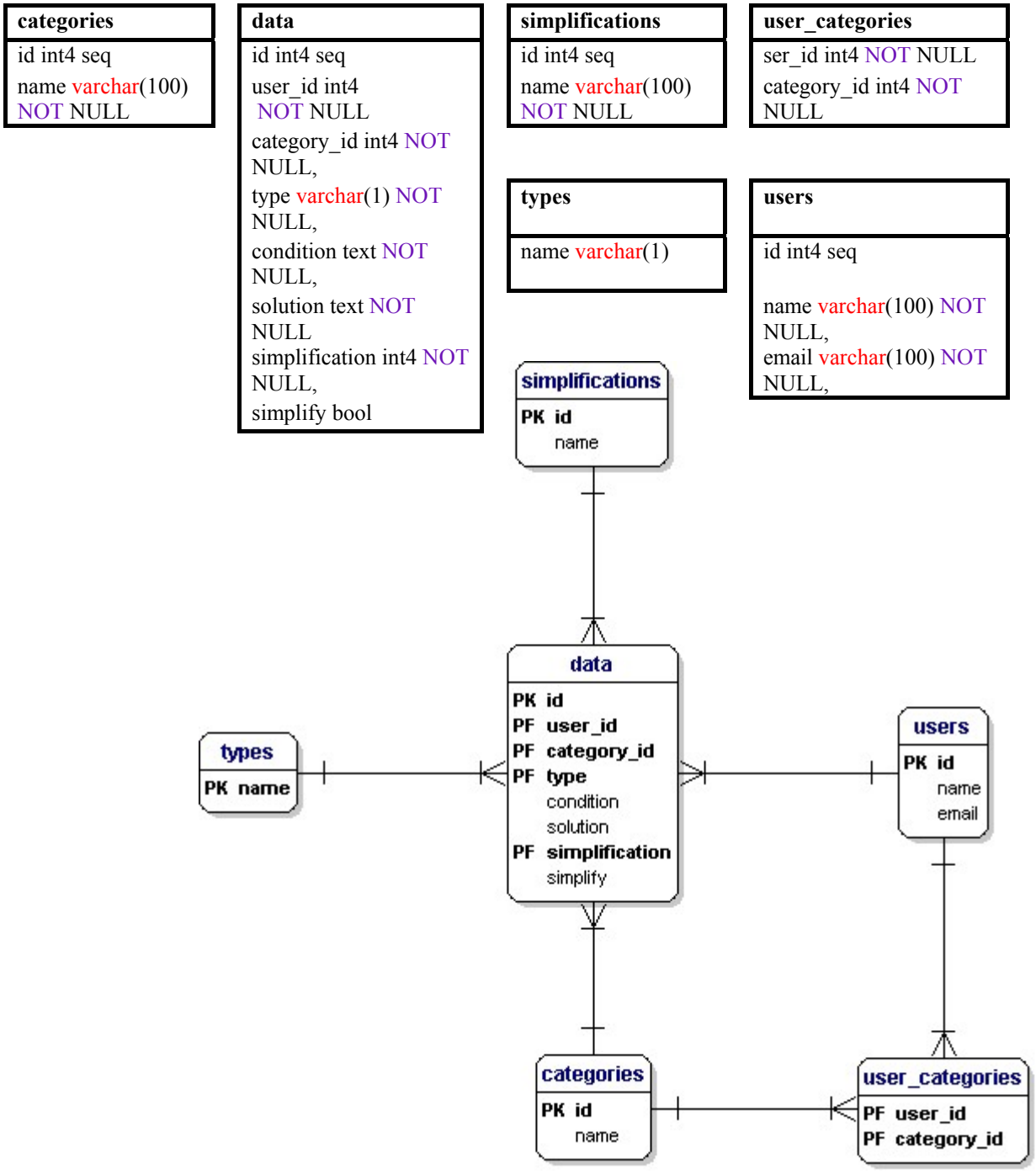
Kompanija IBM platina panašios paskirties „Informix Universal Server“ duomenų bazių tarnybinę stotį ar jų kompleksą, tačiau vien „Informix Standard Engine (SE)“ be papildomų priedų ir geležies licenzija 1000- iui prisijungimų kainuoja begalo didelius pinigus. Šis produktas universalus, įdiegus papildomus modulius galima papildomai paleisti DNS tarnybinę stotį, toks lankstumas būdingas daugumai Unix tipo operacinėms sistemoms. „Informix“ naudoja standartinę „SUN Solaris 2.6“ platformą[26]. Tačiau ne mažesnį lankstumą, patikimumą ir spartą gali pasiūlyti atviro kodo nemokamai platinamos GNU/Linux distribucijos „Debian“, „Slackware“, „ASP Linux“ ir kt.

6.2.1 Duomenų bazės struktūra

Duomenų bazę sudaro šešios pSQL lentelės(6.2.1. lentelė), kurios susietos tarpusavyje pagal 6.2.1pav. parodytą schemą.

6.2.1 lentelė

categories	Vartotojų kategorijų lentelė
data	Pagrindinė duomenų saugojimo lentelė
simplifications	Dinaminių prastinimo bibliotekų lentelė
types	Darbo rėžimų tipai
user_categories	Vartotojų sąryšio su kategorijomis lentelė
users	Vartotojai



6.2.1. pav. ŽB struktūrinė schema ir lentelės

6.3 ZeosLib komponento pritaikymas

ZeosLib – tai atviro kodo projektas, naudojantis didelio našumo technologijas darbui su giminingomis duomenų bazių sistemomis. Pagrindinis ZeosLib bruožas – tas , kad naudojamas universalus, tiesioginis priėjimas prie SQL duomenų bazių tarnybinių stočių tarp dviejų pasaulyje populiariausių operacinių sistemų- “Windows” ir “Linux”. Projektas platinamas pagal LPGL licenziją, kas leidžia naudoti ir platinti produktą nemokamai net komerciniais tikslais[27].

Šiuo momentu ZeosLib suderinamas su:

- Delphi 4 - 7, Kylix 1 - 3, C++ Builder 4 - 6
- MySQL 3.20 - 4.1, PostgreSQL 6.5 - 7.3, Firebird 1.0 - 1.5
- InterBase 5.0 - 7.5, MS SQL 7 - 2000, Sybase ASE 12.0 - 12.5
- Active Data Objects (ADO) Bridge
- IBM DB2 и Oracle

Kuriant automatizuotą teorinių tyrimų terpę buvo pasirinktas ZeosDBO komponentas. Tai supaprastino programos galimybes naudoti nutolusį žinių bazės serverį. Priešingu atveju kiekvienas vartotojas atsisiuntęs programą turėtų sukonfigūruoti ODBC(6.3.1 pav.) . Tai užtrunka laiko ir yra nepatogu, nes kiekvieną kartą startuojant programos aplikacijai ODBC autorizuojasi su nutolusia tarnybine stotimi, bei siunčiant ir priimant užklausas dalyvauja kaip tarpininkas. Tai neleidžia pasiekti gerų reikalavimų vykdymo charakteristikoms(ž.r. 3.2.3). Beja naudojant žemesnių pakopų nei “Windows XP” operacines sistemas būtų privaloma papildomai įsidiegti “Microsoft ODBC” palaikymą. Iš to seka, kad pablogėtų reikalavimai veikimo sąlygoms(ž.r. 3.2.4)

6.3.1. pav

Iš “ZeosLib” projekto automatizuotai teorinių tyrimų aplinkai kurti pritaikyti ir panaudoti komponentai:

 (ZConnection) – naudojamas sesijos su pSQL duomenų baze užmezgimui.

 (ZQuery) – naudojamas siųsti tranzakcijoms per “Zconnection” užmegztą ryšio kanalą.

7. TESTAVIMAI

7.1. Testas 1

Tiesinių lygčių sprendinių tinkamumo tyrimas

Tiesines lygtis bandysime išspręsti simboliniu pavidalu naudojant automatizuotą teorinių tyrimų terpę. Tyrimams pasirenkamos skirtingos reiškinių prastinimo bibliotekos.

Sudarome rezultato tinkamumo kriterijus:

7.1.1 lentelė

Įvertinimo reikšmė	Tinkamumas procentais(%)	Paaiškinimas
0	0	Rezultatas visiškai neatitinka tikrojo ar nepavyko jo gauti.
1	=<30	Rezultatas dalinai teisingas, tačiau yra grubių prastinimo klaidų.
2	=<55	Rezultatas teisingas, tačiau reikalingi dideli reiškinių pertvarkymai.
3	=<75	Rezultatas teisingas, tačiau trūksta nedidelių reiškinių pertvarkymų.
4	>=90	Rezultatas visiškai atitinką tikrąjį arba yra labai artimas.

7.1.1 bandymas

```
duota:
a*x1+b*x2+a*x2+x1*x2+b*x2-x1*x2*a=b+x1-x2*x1
rask: x1
```

Simbolic.dll

```
b - x2 * ( 2 * b + a )
----------------------
 a + x2 * ( 2 - a ) - 1
```

Yacas.dll

```
/ b * x2 + a * x2 + b * x2 - b \
-| --------------------------- |
\   a + x2 - x2 * a - 1 + x2   /
```

Yacas+Simplify()

```
-2 * b * x2 + b - x2 * a
------------------------
  2 * x2 - x2 * a + a - 1
```

Simboliai

```
/ 2 * b * x2 + a * x2 - b \
-| ---------------------- |
\   a + 2 * x2 - x2 * a   /
```

Mathcad 2000 Professional

$$\text{Find}(x1) \rightarrow \frac{(2 \cdot b \cdot x2 + a \cdot x2 - b)}{(-a - 2 \cdot x2 + a \cdot x2 + 1)}$$

7.1.2 lentelė

Įvertinimas	**Biblioteka**
4	Simbolic.dll (Interpreter)
3	Yacas.dll
4	Yacas.dll+Simplify()
4	Simboliai.dll

7.1.2 bandymas

```
duota:
a*x1+b*x2=-b*x1-x2+a
rask: x1
```

Simbolic.dll

$$\frac{a - x2 - b * x2}{a + b}$$

Yacas.dll

$$\left(\frac{b * x2 + x2 - a}{a + b}\right)$$

Yacas+Simplify()

$$\frac{a - x2 - b * x2}{b + a}$$

Simboliai.dll

$$\left(\frac{b * x2 + x2 - a}{a + b}\right)$$

Mathcad 2000 Professional

$$\text{Find}(x1) \rightarrow \frac{(a - b \cdot x2 - x2)}{(b + a)}$$

7.1.3 lentelė

Įvertinimas	Biblioteka
4	Simbolic.dll (Interpreter)
3	Yacas.dll
4	Yacas.dll+Simplify()
3	Simboliai.dll

7.1.3 bandymas

```
duota:
uin*s*c1-u2*s*c1+s=u2/r1-uex/r1+u2*s*c2-u1*s*c2
rask: u1
```

#symbolic.dll

$$u1 = \frac{s * u2 * (c1 + c2) - c1 * s * uin - s}{c2 * s} + \frac{u2 - uex}{c2 * r1 * s}$$

#yacas.dll

$$u1 = \left(\frac{uin * s * c1 - u2 * s * c1 + s - \frac{u2}{r1} + \frac{uex}{r1} - u2 * s * c2}{s * c2}\right)$$

#yacas.dll+Simplify()

$$u1 = \frac{s * c1 * r1 * u2 + s * r1 * u2 * c2 - s * r1 + u2 - uex - uin * s * c1 * r1}{s * r1 * c2}$$

#simboliai.dll

```
                  -1                  -1     -1          -1    -1
u1=   u2 * c1 * c2   - uin * c1 * c2   - c2   + u2 * r1   * s   *

  -1           -1     -1     -1
c2   - uex * r1   * s   * c2   + u2
```

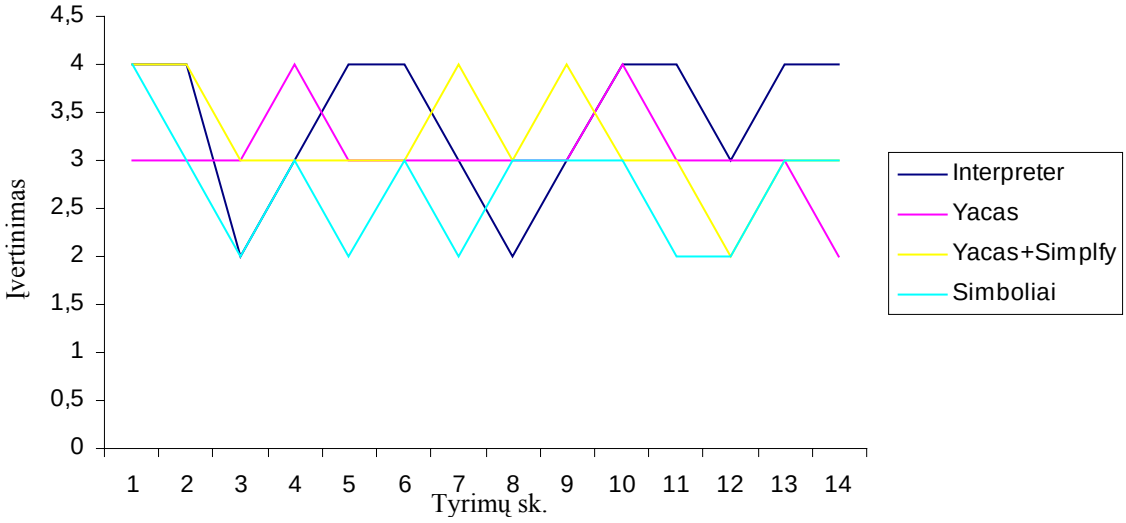
Mathcad 2000 Professional

$$\text{Find}(u1) \rightarrow \frac{(-uin \cdot s \cdot c1 \cdot r1 + u2 \cdot s \cdot c1 \cdot r1 - s \cdot r1 + u2 - uex + u2 \cdot s \cdot c2 \cdot r1)}{(r1 \cdot s \cdot c2)}$$

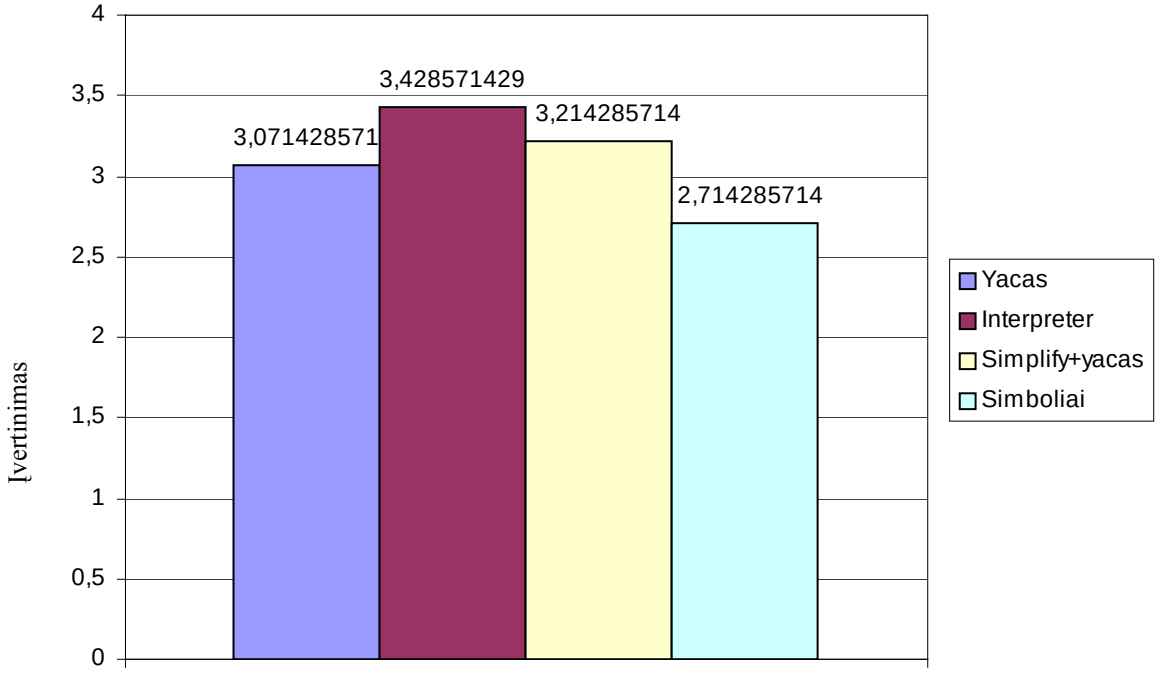
7.1.4 lentelė

Įvertinimas	**Biblioteka**
2	Simbolic.dll (Interpreter)
3	Yacas.dll
3	Yacas.dll+Simplify()
2	Simboliai.dll

Analogiškai atliekame tolimesnius tyrimus spręsdami pavienes lygtis. Ir gautus tyrimų duomenis atvaizduojame grafiškai(7.1.1 pav.).



7.1.1 pav. Tyrimų rezultatai



7.1.2 pav. Tyrimų įvertinimų vidurkiai

Lygčių tyrimų išvados:

Atlikus tiesinių lygčių tyrimus iš gautų tyrimų duomenų(7.1.1 pav.) sunku pastebėti, apie vienos ar kitos dinaminės bibliotekos pranašumą, todėl dar papildomai pateikiame rezultatų įvertinimų vidurkių histogramą(7.1.2 pav.), kur matomas nedidelis "Interpreter" pranašumas, o "yacas" funkcija "Simplify()" nepasižymi labai didele nauda skaičiuojant pavienes tiesines lygtis, tačiau prastina labai panašiai kaip ir Mathcad 2000 Professional. Mažiausio įvertinimo sulaukė "simboliai.dll", tikriausiai dėl to, kad per dažnai naudojamas neigiamas laipsnis. Be to paaiškėjo kad įvertinimo rodiklius stipriai įtakojo ilgesnės ir sudėtingesnės lygtys. Iš to seka, kad sprendžiant ilgas ir sudėtingas tiesines lygtis neverta visada pasitikėti viena kuria tai dinamine biblioteka.

7.2. Testas 2

Tiesinių lygčių sistemų sprendinių tinkamumo tyrimas

Tiesines lygčių sistemas bandysime išspręsti simboliniu pavidalu naudojant automatizuotą teorinių tyrimų terpę. Tyrimams pasirenkamos skirtingos reiškinių prastinimo bibliotekos.
Sudarome rezultato tinkamumo kriterijus:

7.2.1 lentelė

Įvertinimo reikšmė	Tinkamumas procentais(%)	Paaiškinimas
0	0	Rezultatas visiškai neatitinka tikrojo ar nepavyko jo gauti.

1	=<30	Rezultatas dalinai teisingas, tačiau yra grubių prastinimo klaidų.
2	=<55	Rezultatas teisingas, tačiau reikalingi dideli reiškinių pertvarkymai.
3	=<75	Rezultatas teisingas, tačiau trūksta nedidelių reiškinių pertvarkymų.
4	>=90	Rezultatas visiškai atitinką tikrąjį arba yra labai artimas.

7.2.1 bandymas

Sudarome lygčių sistemą su paskutine pertekline lygtimi.

```
duota:
x1 + x2 - 3*x3 = -1
2*x1 + x2 - 2*x3 = 1
x1 + x2+x3 = 3
x1 + 3*x2 - 3*x3 = 1
rask: x1,x2,x3
```

Simbolic.dll	Yacas.dll	Yacas+Simplify()	Simboliai.dll	Mathcad 2000 Prof
x1=1 x2=1 x3=1	x1=1 x2=1 x3=1	x1=1 x2=1 x3=1	# Lygčių sistema neturi sprendinių.	$\text{Find}(x1, x2, x3) \rightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

Nusprendėme patikrinti, kodėl paskutinioji biblioteka neįveikė duotosios užduoties. Iš pagrindinio meniu pasirenkame Nustatymai->Derinimas. Atsiradus “Test” meniu punktui sudėliojame varneles ant: Lygčių statusas ir nariai; koeficientų matrica; tik Gauso matricos formavimas. Apačioje pastebime neteisingai suformuotą koeficientų matricą(7.2.1 pav.).

x1	x2	x3	
1	1	-3	0
2	1	-2	1
1	1	1	3
1	3	-3	1

7.2.1 pav. Koeficientų matrica

Matome neteisingą koeficientą. Pirmosios lygties laisvasis narys yra “-1”. Atidarome programos žurnalą(AI.log) bandydami surasti “simboliai.dll” padarytą klaidą. Randame eilutes.

before: (0) - (1)

after : 0

kur,

before – automatizuotos terpės sprendimo reiškinys, kuris paduodamas išorinei dinaminei bibliotekai apdoroti.

after - rezultatas, kuris grąžinamas automatizuotai terpei atgal po apdorojimo.

Deja, pastebime akivaizdžią reiškinių prastinimo klaidą, dėl kurios tolimesnis lygčių sprendimas butų klaidingas arba negalimas. Todėl programa praneša apie negalimą tokios lygčių sistemos sprendimą.

7.2.2 lentelė

Įvertinimas	**Biblioteka**
4	Simbolic.dll (Interpreter)
4	Yacas.dll
4	Yacas.dll+Simplify()
0	Simboliai.dll

7.2.2 bandymas

```
duota:
x1+2*x2+3*x3+4*x4=5
2*x1+x2+2*x3+3*x4=1
3*x1+2*x2+x3+2*x4=1
4*x1+3*x2+2*x3+x4=-5
rask: x1,x2,x3,x4
```

7.2.3 lentelė

Simbolic.dll	Yacas.dll	Yacas+Simplify()	Simboliai.dll	Mathcad 2000 Professional
x1= **-49999999999999** **---------------** **25000000000000** **x2=** **99999999999999** **--------------** **50000000000000** **x3=-3** **x4=3**	**x1=-2.** **x2=2.** **x3=-3.** **x4=3**	**x1=-2.** **x2=2.** **x3=-3.** **x4=3**	**x1=-2.5** **x2=2.3** **x3=-0.3** **x4=1.2**	$\text{Find}(x1, x2, x3, x4) \rightarrow \begin{pmatrix} -2 \\ 2 \\ -3 \\ 3 \end{pmatrix}$

7.2.4 lentelė

Įvertinimas	**Biblioteka**
3	Simbolic.dll (Interpreter)
4	Yacas.dll
4	Yacas.dll+Simplify()
1	Simboliai.dll

7.2.3 bandymas

```
duota:
x3=x1+a*x2
x1+x2-x3=a
x2-x3+x2=a*x1
```

```
rask: x1,x2,x3
```

7.2.5 lentelė

Simbolic.dll	Yacas.dll

x1=
`    /  2     \     2`
`a * \ a  + 2 /    a`
`-------------- - -----`
`         2       1 - a`
`    1 - a`
 	
x2=
`           a`
`         -----`
`         1 - a`
 	
x3=
`        /  2     \`
`    a * \ a  + 2 /`
`    --------------`
`          2`
`         1 - a`
	x1=
	`   /  2     \              2`
	`   \ a  + 2 / * a         a`
	`--------------------- - -----`
	`( 1 - a ) * ( a + 1 )   1 - a`
	
	x2=
	`              a`
	`            -----`
	`            1 - a`
	
	x3=
	`        /  2     \`
	`        \ a  + 2 / * a`
	`    ----------------------`
	`    ( 1 - a ) * ( a + 1 )`

7.2.5 lentelės tesinys

Yacas+Simplify()	Simboliai.dll

x1=
`  3         2`
`a  - 3 * a  + 2 * a`
`------------------`
`   3    2`
`  a  - a  - a + 1`
	Pakibo prastinant
	2+a^2-((2-a)/(2-a))*(2+a^2)

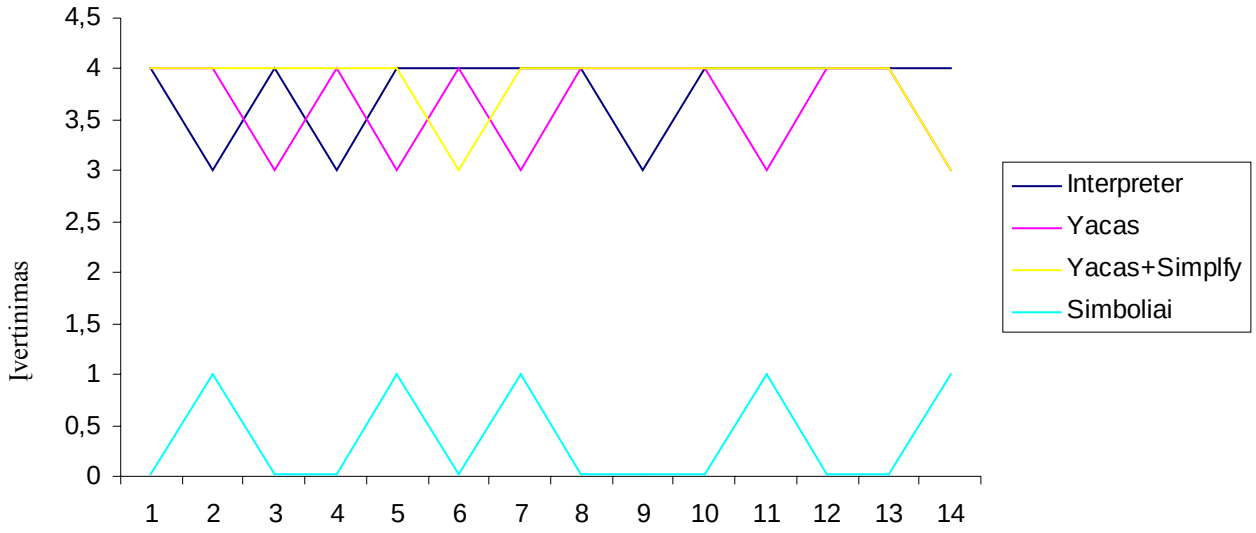
	Mathcad 2000 Prof

x2=
`           a`
`         -----`
`         1 - a`
 	
x3=
`        /  2     \`
`    a * \ a  + 2 /`
`    --------------`
`          2`
`         1 - a`
	$\text{Find}(x1, x2, x3) \rightarrow \begin{bmatrix} (a-2)\cdot\dfrac{a}{\left(-1+a^2\right)} \\ \dfrac{-a}{(-1+a)} \\ -a\cdot\dfrac{\left(2+a^2\right)}{\left(-1+a^2\right)} \end{bmatrix}$

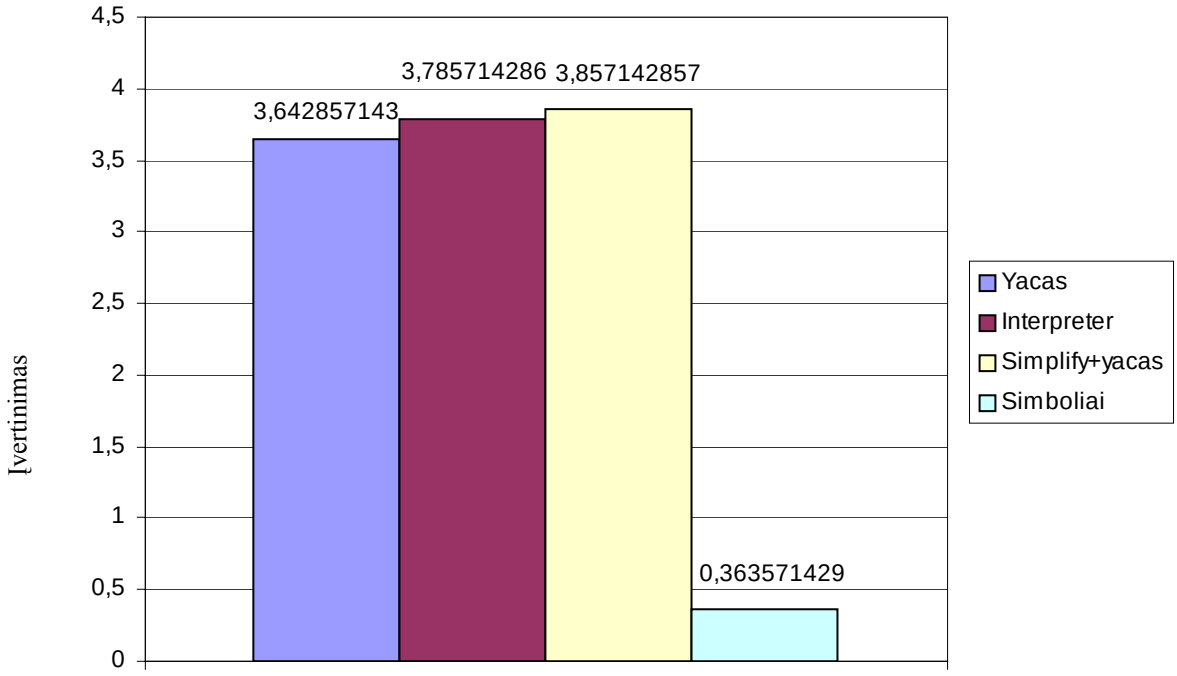
7.2.6 lentelė

Įvertinimas	**Biblioteka**
4	Simbolic.dll (Interpreter)
3	Yacas.dll
4	Yacas.dll+Simplify()
0	Simboliai.dll

Analogiškai atliekame tolimesnius tyrimus. Gautus rezultatus atvaizduojame grafiškai.(7.3,7.4 pav)



7.2.3 pav. Tyrimų rezultatai



7.2.4 pav. Tyrimų įvertinimų vidurkiai

Lyg. Sistemų tyrimų išvados:

Atlikus tyrimus pasirodo, kad simboliai.dll biblioteka sunkiai tvarkosi su sudėtingais ir ilgais lygčių sistemų reiškiniais bei išaiškėjo ir keletas šios bibliotekos atsitiktinių klaidų(7.2.1 bandymas).Visos kitos bibliotekos labai panašiai pateikia rezultatus. Gal kiek kitaip interpretuoja simbolic.dll, tačiau galutinis rezultatas atitinka. Lygčių sistemų testavimuose Mathcad 2000 Prof. dėja nepavyko pralenkti Yacas+Simplify(). Šis duetas puikiai atliko reiškinių prastinimo vaidmenį.

7.3 Testas 3

Elektrinių filtrų perdavimo f-jų skaičiavimo tyrimas

Pagal pateiktas filtrų principines schemas sudarysime tiesinių lygčių sistemas kurias bandysime išspręsti simboliniu pavidalu naudojant automatizuotą teorinių tyrimų terpę ir iš gautos perdavimo funkcijos nustatysime filtro eilę. Tyrimams pasirenkamos skirtingos reiškinių prastinimo bibliotekos.

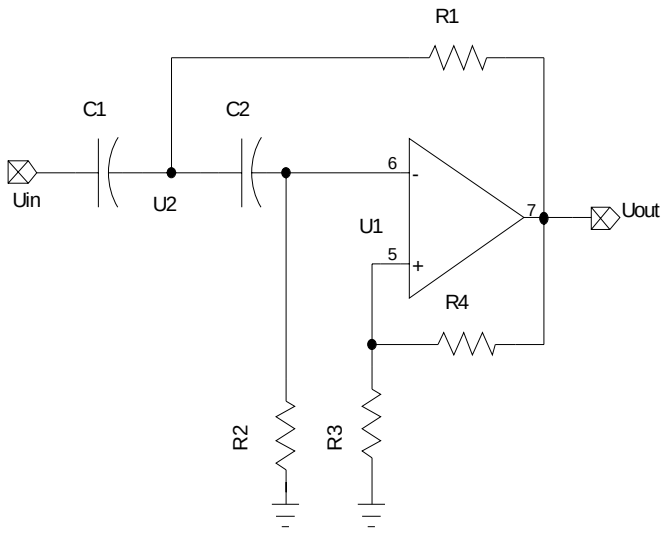
Sudarome rezultato tinkamumo kriterijus:

7.3.1 lentelė

Įvertinimo reikšmė	Tinkamumas procentais(%)	Paaiškinimas
0	0	Rezultatas visiškai neatitinka tikrojo ar nepavyko jo gauti.
1	=<30	Rezultatas dalinai teisingas, tačiau yra grubių prastinimo klaidų.

2	=<55	Rezultatas teisingas, tačiau reikalingi dideli reiškinių pertvarkymai.
3	=<75	Rezultatas teisingas, tačiau trūksta nedidelių reiškinių pertvarkymų.
4	>=90	Rezultatas visiškai atitinką tikrąjį arba yra labai artimas.

7.3.1 bandymas



7.3.1 pav. Selleno-Kėjaus aukšto dažnio filtras

Sudarome lygčių sistemą:

```
duota:
u1=uex*r3/(r3+r4)
uin*s*c1-u2*s*c1=u2/r1-uex/r1+u2*s*c2-u1*s*c2
u2*s*c2-u1*s*c2=u1/r2
rask: u1, u2,uex
```

1) Simbolic.dll (Interpreter)

uex=

$$\left(\left(r1 \cdot s \cdot (c2 + c1) + 1\right) \cdot \left(c2 \cdot s \cdot \frac{1}{r1 \cdot s \cdot (c2 + c1) + 1} - \frac{c1 \cdot c2 \cdot r1 \cdot uin \cdot s^2}{\left(r1 \cdot s \cdot (c2 + c1) + 1\right) \cdot (r3 + r4)}\right) - r3 \cdot \left(\frac{c2 \cdot s}{r3 + r4} + \frac{1}{r2 \cdot (r3 + r4)}\right)\right)$$

(7.3.1)

2)Yacas.dll

$$uex = \frac{\left(\frac{s \cdot c2 \cdot uin \cdot s \cdot c1}{\frac{1}{s \cdot c2 + \frac{1}{r1} + s \cdot c1}}\right) \cdot \left(\frac{1}{r1} + \frac{s \cdot c2 \cdot r3}{r3 + r4}\right) + \frac{1}{r2} + s \cdot c2}{s \cdot c2 \cdot \left(\frac{1}{r1} + \frac{s \cdot c2 \cdot r3}{r3 + r4}\right)} \cdot r3 \qquad (7.3.2)$$

```
                1                          r3 + r4
      s * c2 + -- + s * c1
                r1
```

3) Yacas.dll + Simplify()

uex=

```
-------------------------------------------------------------
          2                                  2
( s * c2 )  * r1 * r2 * r3 * r4 + ( s * c2 )  * r1 * r2 * r4


-------------------------------------------------------------
2   /                2                                  2 \
  - \ ( s * c2 * r1 )  * r3 * r4 + ( s * c2 * r1 * r3 )  / +


-------------------------------------------------------------
       2               2     2        2               2
 -2 * s  * c2 * c1 * r1  * r3  - 2 * s  * c2 * c1 * r1  * r3

                             2
                            s  * c2 * uin * c1 * r1 * r2 *
-------------------------------------------------------------
         2                           2    2
 * r4 + s  * c2 * c1 * r1 * r2 * r4  - s  * c2 * c1 * r1 * r

 /                                            2
 \ -2 * s * c2 * r1 * r3 * r4 - s * c2 * r1 * r3  - s * c2 *
-------------------------------------------------------------
      2   /                2
2 * r3  - \ ( s * c1 * r1 )  * r3 * r4 + ( s * c1 * r1 * r3

        2                                                 2
 r1 * r4  + -2 * s * c1 * r1 * r3 * r4 - s * c1 * r1 * r3  -
-------------------------------------------------------------
 2 \   /  3          2     2                    3          2
)  / - \ s  * c2 * c1  * r1  * r2 * r3 * r4 + s  * c2 * c1

                2                     2     2 \
 s * c1 * r1 * r4  + -2 * r3 * r4 - r3  - r4  /
-------------------------------------------------------------
    2          2    3     2          2                     3
* r1  * r2 * r3  + s  * c2  * c1 * r1  * r2 * r3 * r4 + s  *


-------------------------------------------------------------
   2          2          2 \                          2
 c2  * c1 * r1  * r2 * r3  / + -2 * s * c2 * r1 * r3  - 2 *


-------------------------------------------------------------

s * c2 * r1 * r3 * r4 + s * c2 * r2 * r3 * r4 + s * c2 * r2


-------------------------------------------------------------
    2                         2
```

(7.3.3)

```
* r4  + -2 * s * c1 * r1 * r3  - 2 * s * c1 * r1 * r3 * r4 -


------------------
 /             2 \
 \ r3 * r4 + r3  /
```

4) Simboliai.dll

Programa pakimba prastinant reiškinį:

0-(-r3/(r3+r4))*(-s*c2-r2^-1) (7.3.4)

5) Mathcad 2000 Professional

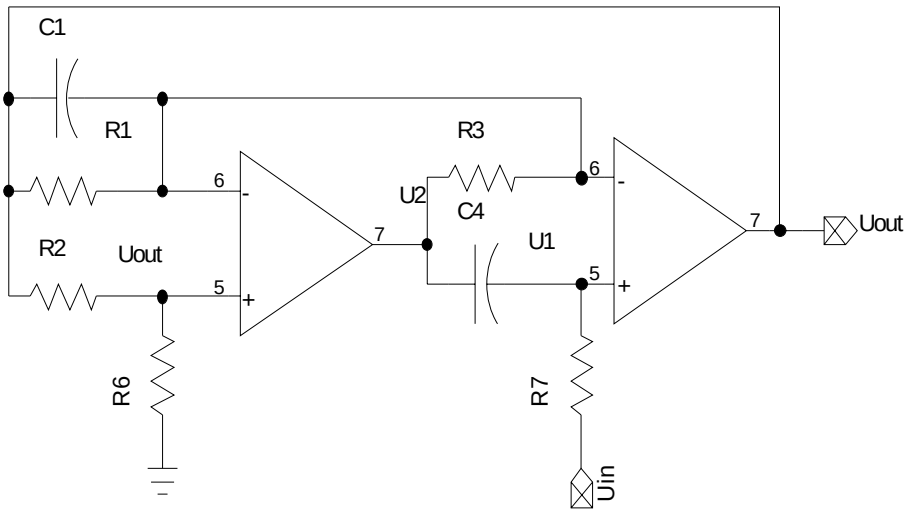
$$\text{Find}(u1, u2, uex) \rightarrow \begin{bmatrix} r3 \cdot uin \cdot s^2 \cdot c1 \cdot r1 \cdot c2 \cdot \dfrac{r2}{\left(-r4 \cdot s \cdot c2 \cdot r2 + r3 \cdot s^2 \cdot c1 \cdot r1 \cdot c2 \cdot r2 + r3 \cdot s \cdot c1 \cdot r1 + r3 + r3 \cdot s \cdot c2 \cdot r1\right)} \\ r3 \cdot uin \cdot s \cdot c1 \cdot \dfrac{r1}{\left(-r4 \cdot s \cdot c2 \cdot r2 + r3 \cdot s^2 \cdot c1 \cdot r1 \cdot c2 \cdot r2 + r3 \cdot s \cdot c1 \cdot r1 + r3 + r3 \cdot s \cdot c2 \cdot r1\right)} \cdot (s \cdot c2 \cdot r2 + 1) \\ uin \cdot s^2 \cdot c1 \cdot r1 \cdot c2 \cdot r2 \cdot \dfrac{(r3 + r4)}{\left(-r4 \cdot s \cdot c2 \cdot r2 + r3 \cdot s^2 \cdot c1 \cdot r1 \cdot c2 \cdot r2 + r3 \cdot s \cdot c1 \cdot r1 + r3 + r3 \cdot s \cdot c2 \cdot r1\right)} \end{bmatrix}$$

7.3.1 formulė geriausiai atitinka elektrinio filtro perdavimo f-ją iš kurios matosi, kad filtras antros eilės. Įvertinę tinkamumo sąlygas sudarome 7.3.2 lentelę.

7.3.2. lentelė

Įvertinimas	**Biblioteka**
3	Simbolic.dll (Interpreter)
2	Yacas.dll
2	Yacas.dll+Simplify()
0	Simboliai.dll

7.3.2 bandymas



7.3.2 pav. ŽDF su dviem operaciniais stiprintuvais

duota:
uex/r2-u1/r2=u1/r6
uin/r7-u1/r7=u1*S*c4-u2*S*c4
uex*S*c1+uex/r1-u1*S*c1-u1/r1=u1/r3-u2/r3
rask: u1,u2,uex

1) Simbolic.dll (Interpreter)

uex=

$$\frac{uin * (r6 + r2)}{c4 * r7 * \left(r3 * s * (r6 + r2) * \left(c1 * s - r6 * \left(\frac{c1 * s}{r6 + r2} + \frac{1}{r1 * (r6 + r2)} + \frac{1}{r3 * (r6 + r2)} \right) + \frac{1}{r1} \right) + r6 * \left(\frac{1}{c4 * r7} + s \right) \right)}$$

(7.3.5)

2) Yacas.dll

uex=

$$\frac{uin}{r7 * S * c4 * r3 * \left(S * c1 + \frac{1}{r1} - \frac{1}{r3} - \frac{1}{\frac{1}{r1} + S * c1 + \frac{1}{r2 * \left(\frac{1}{r6} + \frac{1}{r2} \right)}} - S * c4 * r3 \right)}$$

(7.3.6)

3) Yacas.dll + Simplify()

uex=

```
                                uin
 ----------------------------------------------------------
      2   2                          2
 r1 * r2  * S  * r3 * r7 * c4 * c1 + r1 * r2 * r6 * S  * r3 *

         /  2                    2 \
  * r1 * \ r2  + 2 * r2 * r6 + r6  /
 ----------------------------------------------------------
                            2    2
  r7 * c4 * c1 + r1 * r2 * r6 + r1 * r6  + r2  * S * r3 * r7



 --------------------------------

 c4 + r2 * r6 * S * r3 * r7 * c4
```

(7.3.7)

4) Simboliai.dll

Programa pakimba prastinant reiškinį:

-r7^-1-S*c4-((-r2^-1-r6^-1)/(-r2^-1-r6^-1))*(-r7^-1-S*c4) (7.3.8)

5) Mathcad 2000 Professional

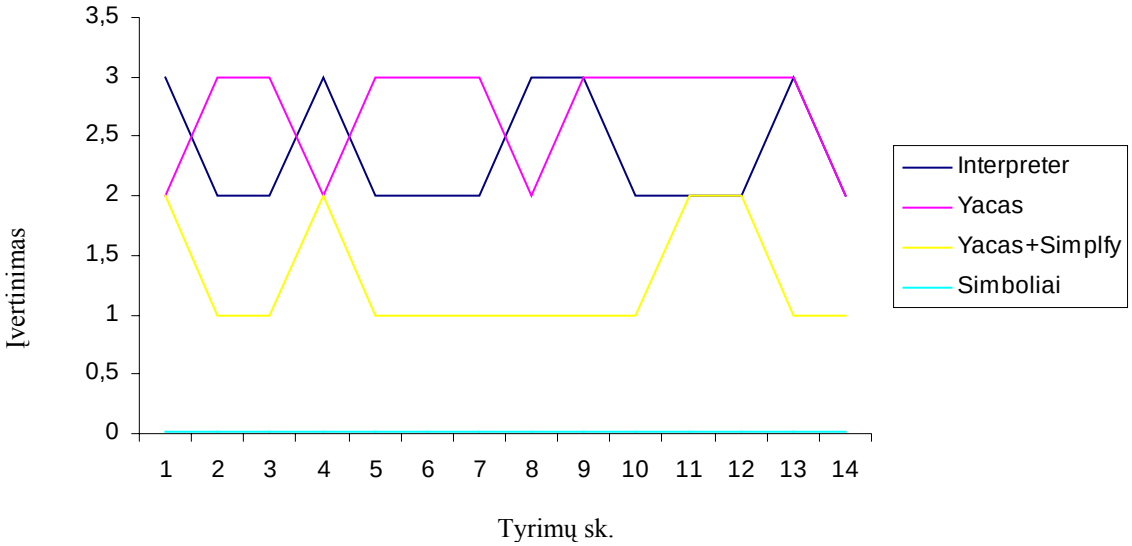
$$\mathrm{Find}(u1,u2,uex) \rightarrow \begin{bmatrix} r1 \cdot r6 \cdot \dfrac{uin}{\left(S^2 \cdot c1 \cdot r1 \cdot r3 \cdot r2 \cdot c4 \cdot r7 + r7 \cdot c4 \cdot S \cdot r2 \cdot r3 + r1 \cdot r6\right)} \\ uin \cdot \dfrac{(-S \cdot c1 \cdot r1 \cdot r3 \cdot r2 - r2 \cdot r3 + r1 \cdot r6)}{\left(S^2 \cdot c1 \cdot r1 \cdot r3 \cdot r2 \cdot c4 \cdot r7 + r7 \cdot c4 \cdot S \cdot r2 \cdot r3 + r1 \cdot r6\right)} \\ r1 \cdot \dfrac{uin}{\left(S^2 \cdot c1 \cdot r1 \cdot r3 \cdot r2 \cdot c4 \cdot r7 + r7 \cdot c4 \cdot S \cdot r2 \cdot r3 + r1 \cdot r6\right)} \cdot (r6 + r2) \end{bmatrix}$$

7.3.6 formulė geriausiai atitinka elektrinio filtro perdavimo f-ją iš kurios matosi, kad filtras antros eilės. Įvertinę tinkamumo sąlygas sudarome 7.3.3 lentelę.

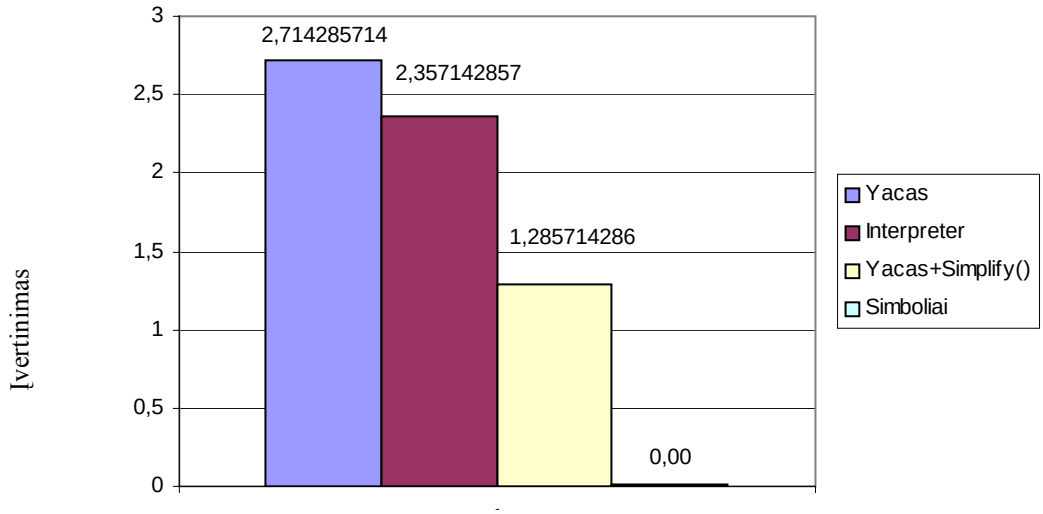
7.3.3. lentelė

Įvertinimas	**Biblioteka**
2	Simbolic.dll (Interpreter)
3	Yacas.dll
1	Yacas.dll+Simplify()
0	Simboliai.dll

Analogiškai atliekame tolimesnius tyrimus su trečios eilės ir kitų tipų filtrais. Ir gautus tyrimų duomenis atvaizduojame grafiškai.



7.3.3 pav. Bibliotekų tinkamumo tyrimo rezultatai



7.3.4 pav. Įvertinimo vidurkiai

Filtrų tyrimų išvados:

Atlikus tyrimus paaiškėjo, kad elektrinių filtrų perdavimo funkcijų skaičiavimams labiausiai tinka “Yacas” biblioteka, nors nuo jos nedaug atsilieka “Interpreter”. Pastebėta, kad “Yacas” funkcija “Simplify()”, kuri skirta reiškinių prastinimui lygčių sistemų sprendime netinkama, o “Simboliai.dll” deja neįveikė nei vieno elektrinių filtrų uždavinio. Mathcad 2000 Professional atsakymuose pateikia vieną bendrą vardiklį per visus nežinomuosius ir papildomai pasikeitimus prie kiekvieno iš jų.

7.4 Testas 4

Simbolinių veiksmų su matricomis tęstinumo tyrimas

Išnagrinėsime simbolinių veiksmų su matricomis tęstinumo galimybes. Tyrimui nebus naudojamos reiškinių prastinimo dinaminės bibliotekos.

Turime dvi matricas A ir B, kur A matrica turi n- eilučių ir m- stulpelių su elementais a[i,j], kad egzistuotų daugyba matricos A eilučių skaičius turi sutapti su matricos B stulpelių skaičiumi t.y m=m bei matricų elementų indeksai turi sutapti j=j.

```
1  duota:
2  A=matrica(n,m,a[i,j])
3  B=matrica(m,k,b[j,z])
4  pateik: F=A*B

6  F = matrica(n, k, f[i, z] = sum({j->m}, (a[i,j] * b[j,z])))        (7.4.1)
```

Gauname naują matricą F kuri turi n- eilučių ir k- stulpelių su naujais elementais f[i,z], kur kiekvienas:

$$f[i,z] = \sum_{m}^{j} a[i,j] \cdot b[j,z] \qquad (7.4.2)$$

Pratęsiame veiksmus su matricomis atlikdami dar vieną daugybą bei sumą. Pasinaudojame jau gauta matrica (7.3.1) ją padauginame iš matricos D

```
8   duota:
9   @6
10  D = matrica(k,p,d[z,k])
11  pateik: G=F*D

13 G = matrica(n,p, g[i, k] = sum({z->k}, (sum(j->m,(a[i,j]*b[j,z])) * d[z,k])))
```

Gauname matricą G, kurios elementai g[i,k]:

$$g[i,k] = \sum_{k}^{z} \sum_{m}^{j} ((a[i,j] \cdot b[j,z]) \cdot d[z,k]) \qquad (7.4.3)$$

Kad atlikti sumą turi sutapti abiejų matricų eilučių ir stulpelių skaičiai.

```
duota:
@13
M = matrica(n,p,m[i,k] = sum({j->t}, (a[i,j] * b[j,k])))
pateik: E=G+M

E = matrica(n, p, e[i, k] = sum(z->k,(sum(j->m,(a[i,j]*b[j,z]))*d[z,k])) +
sum(j->t,(a[i,j]*b[j,k])))
```

Gauname matricą E su elementais e[i,k]:

$$e[i,k] = \sum_{k}^{z}(\sum_{m}^{j}(a[i,j]\cdot b[j,z])\cdot d[z,k]) + \sum_{t}^{j} a[i,j]\cdot b[j,k] \quad (7.4.4)$$

Tyrimų išvada:

Automatizuota teorinių tyrimų terpė naudojanti simbolinius skaičiavimus ir žinių bazes, puikiai vykdo simbolinius veiksmų su matricomis tęstinumo skaičiavimus.

8. DARBO APIBENDRINIMAS IR ATEITIES VIZIJA

Testavimai parodė kuriems automatizuotos terpės darbo rėžimams kokia dinaminė biblioteka geriausiai prastina reiškinius. Iš 7.1 pav. matosi, kad pavienių lygčių sprendime geriausiai pasirodė Interpreter(Simbolic.dll). Tuo tarpu lygčių sistemų sprendime pirmavo Yacas funkcija Simplify(), o elektrinių filtrų perdavimo funkcijos skaičiavimuose dominavo Yacas(8.1 lentelė).

8.1 lentelė

Simbolic.dll	Yacas+Simplify()	Yacas
Lygtys	Lygčių sistemos	Elektriniai filtrai

Tyrimų eigoje paaiškėjo, kad esama simboliai.dll versija gali būti naudojama tik pavienių lygčių sprendime, tačiau sprendimu nevertėtų labai pasitikėti nes biblioteka padaro grubių prastinimo klaidų, dėlto atliekant skaičiavimus vertėtų nepamiršti įsijungti “DEBUG” langų ir sprendimus pakartoti su kitomis bibliotekomis.

Automatizuota teorinių tyrimų terpė naudojanti simbolinius skaičiavimus ir žinių bazes,- puikus įrankis kiekvienam norinčiam prie tyrimų prisidėti ar juos atlikti. Šiuo metu terpė skaičiuoja simboliškai ir skaitiškai veiksmus su lygtimis jų sistemomis bei atlieka simbolinius veiksmus su matricomis. Turi galimybę informaciją saugoti ir užkrauti “XML” ir “TXT” formatais, bei gali visus skaičiavimo rezultatus ir sąlygas automatiškai talpinti į žinių bazių serverį ir pagal automatinę vartotojo veiksmų sekimo metodiką gražinti alternatyvius paieškos rezultatus. Dabartinė programos versija disponuoja trimis išorinėmis reiškinių prastinimo bibliotekomis ir viena papildoma prastinimo funkcija. Programa pagal saugomą informaciją ir duomenų formatą(XML) automatiškai generuoja “XSL” bylą, kuri leidžia XML formatu išsaugotą informaciją atvaizduoti interneto naršyklei(pvz. IExplorer). Naudojant ŽBS, kiekvienas nutolęs vartotojas turi galimybę kurti kategorijas pagal jo paties atliekamų tyrimų specifiką. Kai pasirodo naujesnė programos versija, programa automatiškai praneša apie tai naudotojui. Kadangi dabar laisvai platinama pirmoji “stable” aplinkos versija, tai neatmetama galimybe ateityje pažaboti daugiau aplinkos darbo rėžimų ir papildomų funkcijų. Matricų ir lygčių klases galima atskirai kompiliuoti bet kokioje platformoje ar operacinėje sistemoje ir pritaikyti bei keisti išeities kodą pagal poreikius. Tai leidžia projektui judėti naujų minčių ir idėjų įgyvendinimo linkme. Naudinga būtų, žinių bazės serverio žinias panaudoti platesnėms reikmėms, o gal net praplėsti žinių turinį, jo įsisavinimo bei atidavimo galimybes. Žinių bazių serveriu galėtų naudotis informaciniai agentai, kur agentas,- *autonominis programinis vienetas, kuris turi priėjimo teisę prie heterogeninių ir geografiškai pasiskirsčiusių šaltinių internete.*

Informacijos įgijimas ir valdymas. Gali suteikti priėjimą prie keleto skirtingų informacijos šaltinių. Perskaito, išpakuoja, analizuoja ir filtruoja duomenis, atnaujina svarbią vartotojui ar kitam

agentui informaciją. Realiai informacijos įgijimas įgauna plačią prasmę, jis atliekamas tiek žinių bazėse, tiek elektroniniuose šaltiniuose.

Aplankykite projektą internete http://ai.macrosoft.lt

IŠVADOS

Vienas kompiuterio privalumų yra tai, kad jis leidžia kiekvienam besimokančiam judėti pirmyn tuo greičiu, kurį lemia jo protinių procesų sparta bei asmeniniai bruožai, auklėjimo ypatumai, įgimtos savybės ir t.t. Versti visus besimokančius judėti pirmyn vienodu greičiu, kaip tai neišvengiamai atsitinka naudojant įprastus mokymo metodus, nėra pats geriausias būdas.

Automatizuota teorinių tyrimų terpė palengvins tyrimų eigą ir analitinius skaičiavimus, leis viešai publikuoti darbo rezultatus bei pakreipti tyrimus naujos idėjos linkme, padės surasti panašių tyrimų ar sprendimų profilio bendraminčių.

Naują žinių bazių sistemos modelį būtina integruoti į tyrimų ir kitokio mokslino pobūdžio programines įrangas, tobulinti rezultatų grąžinimo bei ieškos mechanizmus. Galbūt tai pirmieji intelektualios aplinkos atsiradimo žingsniai.

Tokia automatizuota teorinių tyrimų terpė plačiausiai taikoma „Grandinių teorijos“ , „Signalų ir sistemų“ bei „Neuronų tinklų“ disciplinose, nors egzistuoja reali galimybė naudoti šią terpę ir kitose disciplinose bei praktiniuose skaičiavimuose.

Galima pasinaudoti komerciniais produktais kaip „Informix Universal Server“, Mathematica, MatLAB, MAPLE ir kt., tačiau tokia programinė įranga nėra specializuota, kelia aukštesnius reikalavimus veikimo sąlygoms(ž.r.3.2.4.1 lentelė), su tokia įranga nepateikiamas išeities kodas, todėl į tokias programas praktiškai neįmanoma integruoti žinių bazių sistemų, jos dažniausiai neturi “DEBUG” langų,- tarpinių rezultatų stebėjimui bei yra brangiai apmokamos. Nemokami produktai labiau specializuoti, lengvai integruojami, skatina kūrybines idėjas ir labiau įkvepia vartotojus dirbančius su laisvai platinama produkcija. Nors naudotojas psichologiniu požiūriu mažiau pasitiki laisvai platinimu produktu, tačiau patyrusiems naudotojams tą kompensuoja galimybė dalyvauti produkto vystyme, įdėti papildomas galimybes.

LITERATŪRA

1. Software Engineering. Ian Sommerville. Addison-Wesley 1992; 1-5
2. http://www.soften.ktu.lt/~pik98/DT/TEORIJA/tema_5.html
3. http://www.maplesoft.com
4. http://www.mathematica.com
5. http://www.sciface.com
6. http://www.mathworks.com
7. http://www.symbolicnet.org
8. http://www.lernnetz-sh.de/opensource/beispiele/yacas/
9. Evans B. L. et al. Learning Signals and Systems with Mathematica //IEEE Transactions on Education. –1993. Vol. 36, No.1. – P. 72-78.
10. Beltzer A. I., Shenkman A. L. Use of Symbolic Computation in Engineering Education //IEEE Transactions on Education. –1995. –Vol. 38, No.2 –P. 177-184
11. Munteanu I., Ioan D. Symbolic Computation with Maple V for Undergraduate Electromagnetics //IEEE Transactions on Education. –2001. –Vol. 44, No.2. –3p.
12. http://www.analog-insydes.de
13. Huelsman L.P. Symbolic Analysis –A Tool for Teaching Undergraduate Circuit Theory //IEEE Transactions on Education. 1996. –Vol. 39, No.2. –P 243-250.
14. Lucheta A., Manetti S., Reatti A. SAPWIN – A Simbolic Simulator as a Support in Electrical Engineering Education //IEEE Transactions on Education. –2001. –Vol. 44, No. 2. –9p.
15. http://nkm.lt/index.phtml?lst=article&action=view_article&id=480
16. http://www.gnu.org/fsf/fsf.html
17. http://www.gnu.org
18. http://www.volere.co.uk/template.htm
19. http://www.soften.ktu.lt/~pik98/DT/TEORIJA/tema_11.html
20. Gonzalez A.J., Douglas D. The Engineering of knowledge-based systems theory and practice. 1993.
21. http://www.how.lt/zinios/2_1.htm
22. Эвальд Гешвинде, Ганс-Юрген Шениг. PostgreSQL. Руководство разработчика и администратора – 2002. ДиаСофт. 20-23p
23. http://advocacy.postgresql.org/
24. http://www.rfc-editor.org/
25. http://www.postgresql.org/docs/aw_pgsql_book/node43.html
26. http://wwwdb.informatik.uni-rostock.de/systems/informix.html

27. http://www.zeoslib.net

28. http://www.toto.ot.lt/iSaviugda/instruction.html

29. G.Misevičius ir kt. Aukštoji matematika. Vadovėlis ir pratybos su kompiuteriu. –1999. Vilnius. TEV. 44-15p.

11. SUTRUMPINIMAI

Trumpinys	Paaiškinimas
IPT	Interneto paslaugų tiekėjas
GNU	General Public License
LPGL	Lesser Gnu Public License
SQL	Manipuliavimo duomenų bazių sistemomis programavimo kalba
pSQL	PostgreSQL
ŽB	Žinių bazė
ŽBS	Žinių bazės serveris
RFC	Request for Comments
ODBC	Open DataBase Connectivity

CD TURINYS

Nr.	Katalogo pavadinimas	Aprašymas	Dydis,Mb
1	Ai	Pagrindinė programa be išeities kodo	3,59
2	Ai_source	Pagrindinė programa su išeities kodu	5,76
3	Help	Naudojimo instrukcija	1,82
4	psql	PSQL duomenų bazės įdiegimo skriptas ir nurodymai	0,117
5	nfo	Magistro darbas “doc” formatu	4,73
6	www	Programos internetinė svetainė	2,56
7	zeos	ZeosLib komponentas	1,32
8	rar	Suarchyvuoti visi katalogai	6,5

Priedas 1

NAUDOJIMO INSTRUKCIJA

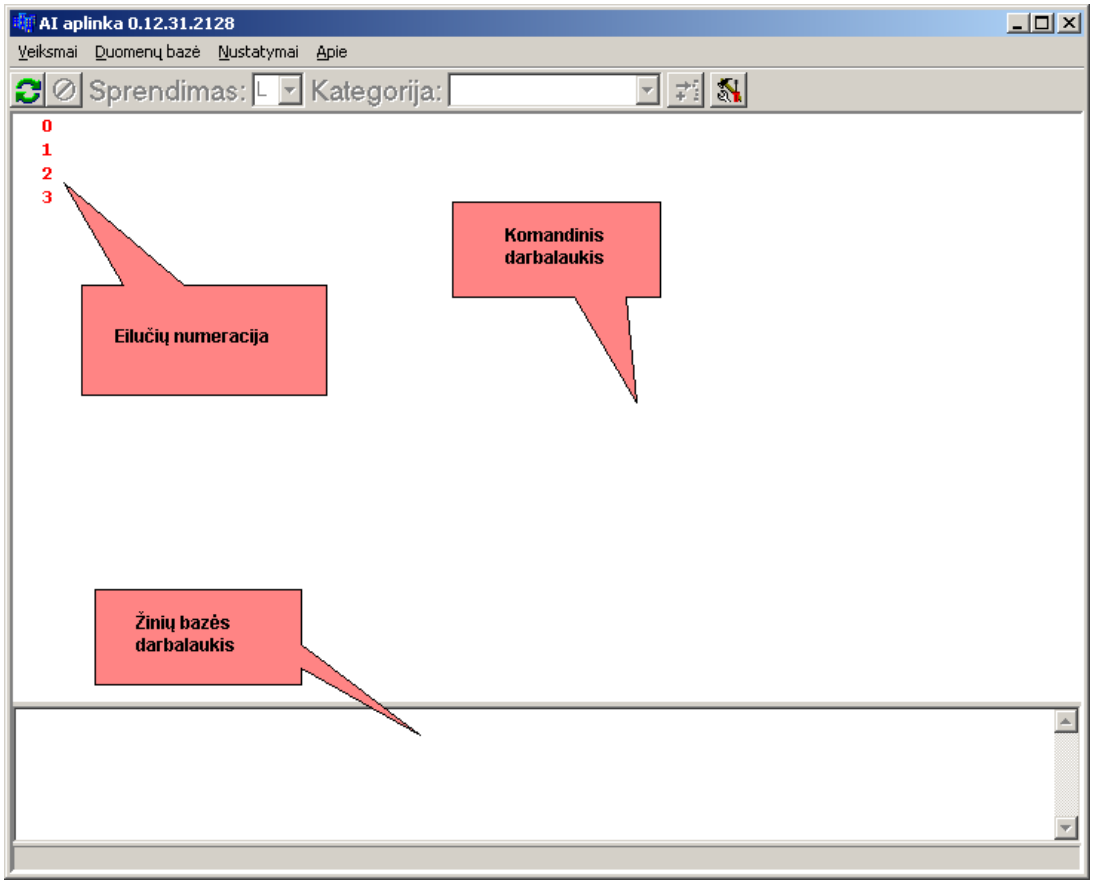
1. Paskirtis

Švietimo kokybinė raida lemia naujų požiūrių į mokymą atsiradimą. Lietuvoje jau siekiama įgyvendinti mokymosi "visą gyvenimą" principą. Mokymą vis labiau pakeičia savimoka, saviugda. Į ugdymo procesą vis labiau įsilieja kompiuteriai, specialios mokymo programos, žaidimai, treniruokliai. Praktika patvirtinta, kad labai gabus studentas sugeba per dvi dienas (!) išmokti kompiuteriu užprogramuotą vienerių metų matematikos kursą.[28] Programa skirta visiems, kurie dažnai susiduria su lygčių, jų sistemų ar matricų simboliniais , analitiniais skaičiavimais. Tačiau ši automatizuota terpė labiau orientuota elektronikos inžinerijos disciplinoms, kaip "Grandinių teorija", Analoginiai ir skaitmeniniai įtaisai", "Neuronų tinklai" ir kt.

Programos naudojimas susideda iš dviejų pagrindinių etapų: paruošimo ir naudojimo.

2. Paruošimas

Programa automatizuota, todėl reikalauja minimalaus pasiruošimo darbui. Pasiruošimas susijęs su interneto ryšių, jei jo nėra galima pradėti dirbti. 2.1 pav. iliustruoja pagrindinį programos langą.



2.1 pav. Darbalaukiai

3. Naudojimas

Pradedant darbą su 2.1 pav. pateikta aplikacija svarbiausia susipažinti su darbalaukiais, meniu ir komandomis. Programoje dirbama dviem darbalaukiais:

1) Komandiniu

2) Žinių bazės

Komandinio kairėje rašoma eilučių numeracija, kuri naudojama jau anksčiau parašytų eilučių panaudojimui. Pvz. gautą rezultatą “10” eilutėje panaudojame vėliau “24” eilutėje parašydami “@” simbolį ir norimos naudoti eilutės skaičių. Esant eilučių pasikeitimams programa automatiškai pakeis eilutės skaičių prie simbolio.

```
5  duota:
6  M = matrica(d,z,m[i,k] = sum({j->p}, (a[i,j] * b[j,k])))
7  A = matrica(z,p,a[k,j])
8  pateik: G=M*A
9
10 G = matrica(d, p, g[i, j] = sum({k->z}, (sum(j->p,(a[i,j]*b[j,k])) *
a[k,j])))
```

```
23 duota:
24 @10
25 D= matrica(p, h, d[j, k])
26 pateik: T=G*D
27

   28  T  =  matrica(d,  h,  t[i,  k]  =  sum({j->p},  (sum(k->z,(sum(j->p,
(a[i,j]*b[j,k]))* a[k,j])) * d[j,k])))
```

Raudonai žymimos nenaudojamos, sintaksės ar sąlygos teisingumo neatitinkančios eilutės.

Pvz.

```
0     duota  # neteisinga sintaksė

1  duota: # teisinga sintaksė
```

Komandinės sąsajos pagrindinės komandos ir sintaksė pateiktos 3.1 lentelėje:

Lentelė 3.1

Lygtys ar jų sistemos		Matricos	
duota:	sąlyga	**duota:**	sąlyga
rask:	rezultatas	**pateik:**	rezultatas
@	tęstinumas	@	tęstinumas
#	komentaras	#	komentaras
		matrica()	matrica
+	sudėtis	+	sudėtis
-	atimtis	-	atimtis
/	dalyba	/	dalyba
*	sandauga	*	Sandauga
^	laipsnis		

Sintaksė supaprastinta siekiant neapkrauti dirbančiojo ir skiriasi tik rezultato išvedimo komandinis žodis. Reikia nepamiršti, kad prie sąlygos ir išvedimo operatorių pridedamas "**:**".

3.1 Kaip išvengti klaidos pranešimo?

Norint išvengti pranešimų apie klaidas vertėtų žinoti, kokios klaidos daromos dažniausiai. 3.1.1, 3.1.2 ir 3.1.3 lentelėse pateikiame klaidų pranešimus ir paaiškinimus :

Lentelė 3.1.1

Lygtys ir jų sistemos	
Klaida	**#Klaidos pranešimas**
Kai norima rasti daugiau šaknų, nei yra lygčių sistemoje.	# Per mažai lygčių.
Kai bandoma spręsti lygtį ir lygtis neturi tokio nežinomojo.	# Nerastas lygtyje nežinomasis.
Kai bandoma spręsti lygčių sistemą ir nė viena tos sistemos lygtis neturi tokio nežinomojo.	# Nerastas lygtyse nežinomasis.
Kai bandoma spręsti lygčių sistemą ir norima rasti kelis tuos pačius nežinomuosius.	# Dubliuoti nežinomieji Rask: eilutėje.
Kai bandoma išspręsti lygčių sistema neturi sprendinių.	# Lygčių sistema neturi sprendinių.
Kai lygtyje/lygčių sistemoje ar žodeliuose „duota" ir „rask" yra klaidų.	# Lygtyse yra klaidų.

Lentelė 3.1.2

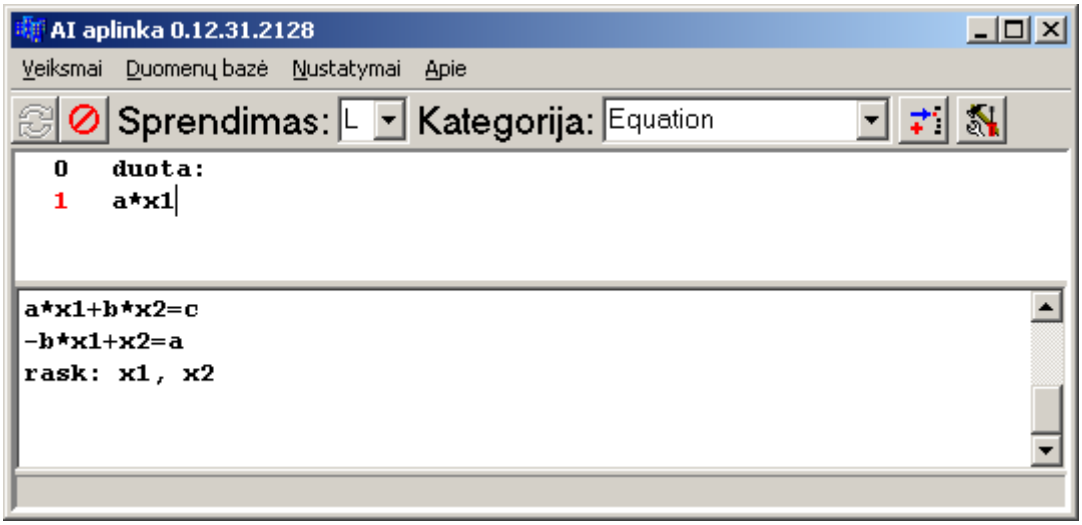
Matricos	
Klaida	**#Klaidos pranešimas**
Kai rezultatų matricos pavadinimas sutampa su sąlygoje duotos matricos pavadinimu.	# Blogas matricos veiksmų rezultatų pavadinimas.
Kaip bandoma atlikti nerealizuotą/nesuprantamą matricų operaciją.	# Nesuprantama operacija.

Kai norima atlikti operaciją, kuriai reikia, kad matricų elementų intervalai sutaptų.	# Matricų galų intervalai nelygus.
Kai norima sudauginti matricas ir atitinkami matricų elementų intervalai nesutampa.	# Matricų galų intervalai netinka sandaugai.
Kai norima atlikti operaciją, kuriai reikia, kad matricų elementų indeksai sutaptų.	# Matricų elementų indeksai nelygus.
Kai norima sudauginti matricas ir atitinkami matricų elementų indeksai nesutampa.	# Matricų elementų indeksai netinka sandaugai.
Kai matricose ar žodeliuose „duota“ ir „pateik“ yra klaidų.	# Matricose yra klaidų.

Lentelė 3.1.3

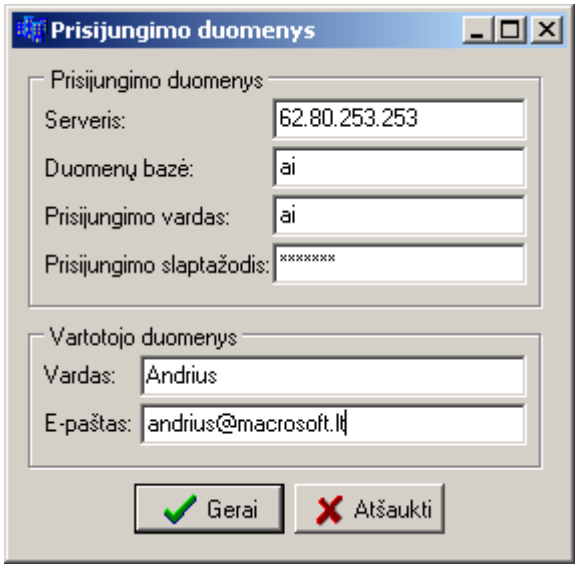
Eilutė yra klaidinga (statuse eilutės numeris paraudonuoja) kai
Trumpesnė nei 3 simboliai
Neturi lygybės simbolio ir tai ne žodeliai: "duota:", "rask:", "pateik:"
Pirmas arba paskutinis simbolis yra lygybės ženklas
Pirmas arba paskutinis simbolis yra daugybos ženklas
Yra daugiau nei vienas lygybės ženklas ir nėra žodelio "matrica"
Yra žodelis "matrica" ir gale nėra ")"

Žinių bazės darbalaukis tarnauja kaip greita pagalba vartotojui ar vykdomų skaičiavimų gidas(3.1pav).



3.1 pav. ŽB darbalaukis

Rašant reiškinius, kai įjungta naudojimosi žinių bazės sistema, po kiekvienos pauzės(kai dvejojama ką rašyti toliau) frontendas kreipiasi į žinių bazę su prašydamas gražinti rezultatą pagal esamą kursoriaus eilutę. Kad įgalinti žinių bazę, reikalinga užpildyti duomenų bazės nustatymų formą(3.2pav.).



3.2 pav. Prisijungimo duomenys

Paskutinio prisijungimo duomenys išlieka. Kaskart įjungus programą būtina užpildyti skiltį “Vartotojo duomenys”. Kaip ir dauguma IP protokolu dirbančių programų automatizuota terpė naudoja autorizaciją. Autorizacija reikalauja prisistatyti žinių bazės tarnybiniai stočiai.

Kol nenaudojama žinių bazė “Sprendimas” ir “Kategorija” yra neaktyvūs.

Kategorija sukuriama pačio vartotojo, atsižvelgiant į atliekamų tyrimus ar skaičiavimus. Kategorija įterpiama naudojant mygtuką. Tai palengvina duomenų rūšiavimą ir paiešką.

Pastaba: jei kitą kartą norėsite išvysti savo kategorijas, būtina suvesti vartotojo vardą ir e.paštą identiškai pirmajam prisijungimui.

Sprendimas skirtas talpinant informaciją į ŽBS priklausomai pagal darbo režimą

“**M**” – matricos

“**L**” – lygtys ir jų sistemos

“Sprendimas” automatiškai detektuojamas pagal rezultatų išvedimo operatorius:

“**pateik:**” – lygtys ir jų sistemos

“**rask:**” - matricos

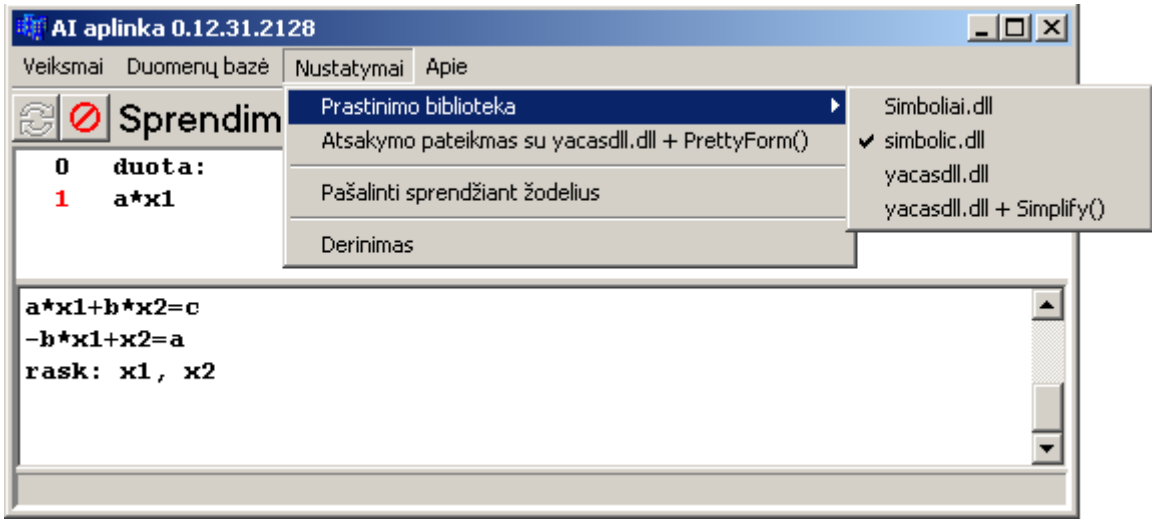
 - Prisijungimas prie ŽBS

 - Atsijungimas nuo ŽBS

 - ŽBS prisijungimo parametrų nustatymai

3.2 Programos rezultato pateikimo nustatymai

Nustatymuose galima pasirinkti rezultato atvaizdavimo forma(3.3 pav. “Atsakymo pateikimas su yacassdll.dll + PrettyForm()). Tokiu atveju rezultatas pateikiamas laužytoje formoje(3.4 pav).



3.3 pav. Rezultato pateikimo nustatymai

```
duota:
a*x1+b*x2=c
-b*x1+x2=a
rask: x1, x2

x1=
        /  2          \
c   b * \ a  + b * c /
- - ------------------
a          /  2     \
       a * \ b  + a /
```

```
x2=
 2
a  + b * c
----------
   2
  b  + a
```

3.4 pav rezultatų pateikimas naudojant “PrettyForm()”

“Prastinimo biblioteka” sudaro importuotų bibliotekų rinkinys:

Simboliai.dll – pačių rašyta reiškinių prastinimo biblioteka.[]

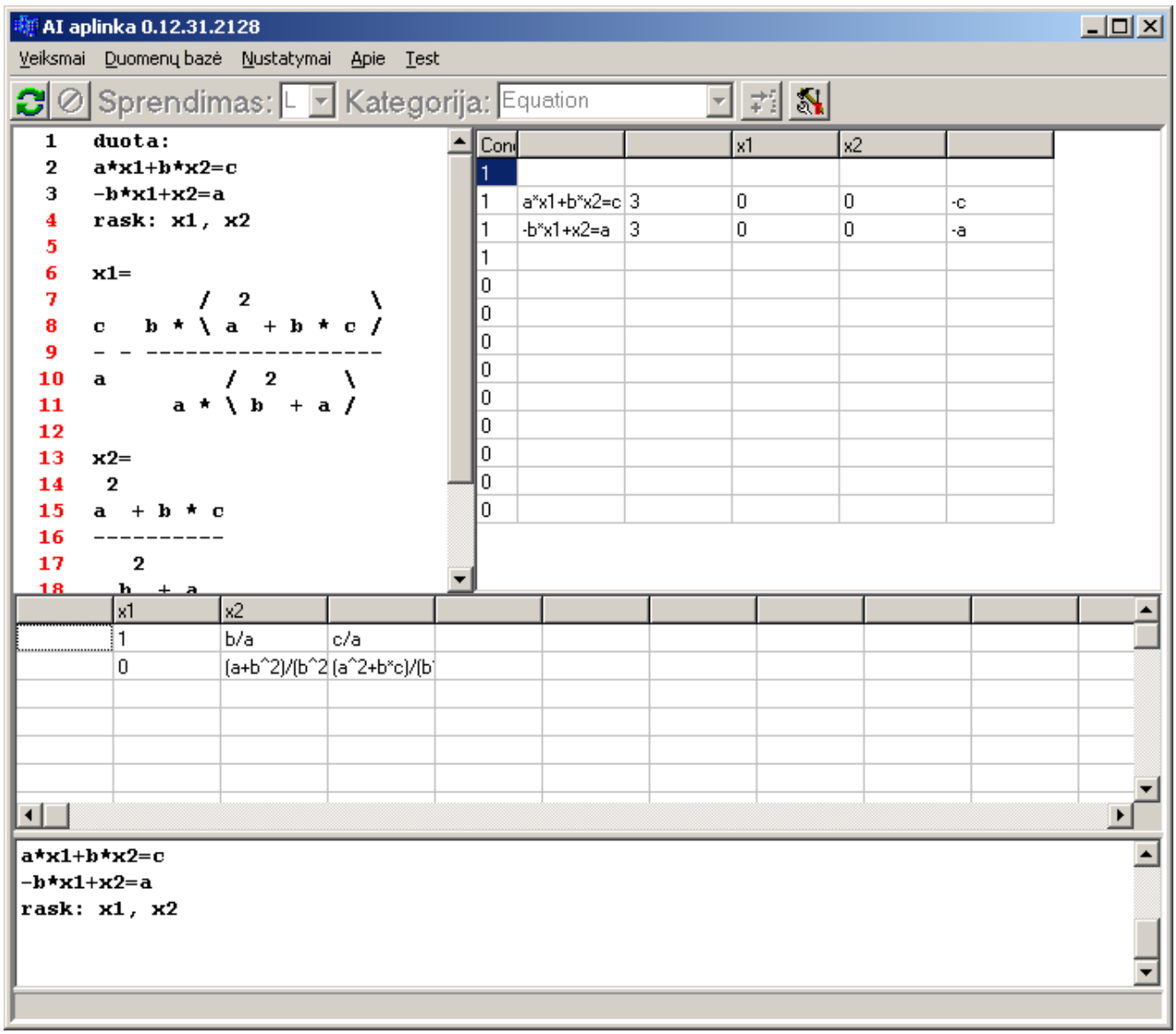
Simbolic.dll - parašyta “Wismar” vokiečių aukštosios mokyklos studentų.[]

Yacas.dll – parašyta olando A.Pinkaus kartu su kitais mokslininkais ir programuotojais.[]

Yacas + Simplify() – naudojama paildoma “*Yacas.dll*” reiškinių prastinimo “*Simplify()*” funkcija.

Kiekvieną iš bibliotekų galima naudoti lygiagrečiai su rezultato pateikimo forma(PrettyForm()).

Nustatymuose galima pasirinkti,- pašalinti sprendžiant žodelius “duota:” ,“rask:”, “pateik:” ar ne. Taipogi yra galimybė pasirinkti derinimo(DEBUG) darbo režimą.



3.5 pav. DEBUG darbo rėžimas

Dešinėje lango pusėje matome lygčių skaidymą į koeficientus, o apačioje Gauso matricos formavimą. Vaizdavimo langus ir jų paskirtį galima keisti dėliojant varneles atsiradusiame “Test” meniu.

Atliktus darbus galima išsaugoti ir atsidaryti “TXT” ar “XML” formatais. Saugant duomenis “XML” sukuriama “XSL” byla tokiu pat pavadinimu. Ši byla leidžia mūsų išsaugotą “XML” peržiūrėti su interneto naršykle pvz. “Microsoft Internet Explorer 6.0”.

Priedas 2

Pateikiame tik lygčių ir matricų klasių išeities kodus.

#---------- Simbolicequations.h ----------

```
#ifndef SIMBOLICEQUATIONS_H
#define SIMBOLICEQUATIONS_H

#include <string>
#include <map>

using namespace std;

/**
 * Simbolinių lygčių ir jų sistemų skaičiavimo klasė.
 *
 * Versija: 0.32
 **/
class SimbolicEquations
{
public:
    // Galimų klaidų sąrašas.
    enum ErrorCodes {
        NONE,                     // Jokių klaidų.
        BAD_EQUATION_FORMAT,      // Blogas lygties formatas.
        NOT_ENOUGH_EQUATION,      // Permažai lygčių.
        UNKNOWN_NOT_FOUND,        // Nerastas lygtyje nežinomasis.
        UNKNOWN_NOT_FOUND_M,      // Nerastas lygtyse nežinomasis.
        DUBLICATED_UNKNOWNS,      // Pasikratojantys rask dalyje nežinomieji.
        SYSTEM_DO_NOT_HAVE_ROOTS  // Lygčių sistema neturi sprendinių.
    };

    // Naudojamų prastinimo bibliotekų sąrašas.
    enum SimplifyLibraries {
        SIMBOLIAI_DLL,         // Simboliai.dll
        SIMBOLIC_DLL,          // simbolic.dll
        YACASDLL_DLL,          // yacasdll.dll
        YACASDLL_DLL_SIMPLIFY  // yacasdll.dll + Simplyfy()
    };

    // Derinimo (DEBUG) duomenų tipų sąrašas.
    enum DebugTypes {
        EQUATIONS,        // Išskaidyta lygčių sitemos matrica.
        GAUSUS_MATRIX,    // Suformuota sprendimui Gauso matrica.
        GAUSUS_SOLUTION  // Išpręsta Gauso matrica.
    };

    // Derinimo (DEBUG) duomenų tipų sąrašas.
    enum DataTypes {
        MIN_X, // Minimali matricos x koordiantė.
        MIN_Y, // Minimali matricos y koordinatė.
        MAX_X, // Maksimali matricos x koordinatė.
        MAX_Y  // Maksimali matricos y koordiantė.
    };

public:
   /**
    * Klasės destruktorius.
    **/
    ~SimbolicEquations();

   /**
    * Klasės konstruktorius.
    **/
    SimbolicEquations();

   /**
    * Išvalo lygčių buferį.
    **/
    void Clear();

   /**
    * Prideda lygtį į lygčių buferį.
    *
    * Parametras: equation - lygtis.
    **/
```

```
 void Add(const string &equation);

/**
 * Atlieka lygties, lygčių sistemos sprendimą.
 *
 * Parametras: radicals - kokias šaknis reikia rasti.
 **/
 void Execute(const string &radicals);

/**
 * Grąžina rezultato eilučių skaičių.
 *
 * Rezultatas: rezultato eilučių skaičius.
 **/
 int AnswerCount() const;

/**
 * Grąžina rastos šaknies rezultatą.
 *
 * Gražina nurodytą rezultato eilutę.
 *
 * Parametras: answerNm rezultato eilutė.
 * Rezultatas: Rastos šaknies rezultatas.
 **/
 string Answer(const int &answerNm);

/**
 * Grąžina paskutinės atliktos operacijos klaidos kodą.
 *
 * Rezultatas: Paskutinės atliktos operacijos klaidos kodas.
 **/
 int Error() const;

/**
 * Nustato ar reikia pateikti ataskymą gražia forma.
 *
 * Parametras: yes - jei true tai taip, jei folase - tai ne.
 **/
 void SetUsePrettyForm(const bool &yes);

/**
 * Nustato kokia prastininmo bilioteką reikia nudoti.
 *
 * Parametras: mode - bibliotekos numeris iš SimplifyLibraries.
 **/
 void SetSimplifyMode(const SimplifyLibraries &mode);

/**
 * Nustato kur ir kokiu pavadinimu reikia išsaugoti log'ą.
 *
 * Parametras: fname - log'o failo vardas.
 **/
 void SetLogFileName(const string &fname);

/**
 * Matricos koordinačių intervalo gavimas.
 *
 * Parametrai: type  - kurios matricos norima sužinoti koordintę;
 *             dtype - kurią matricos koordinatę norima sužinoti.
 * Rezultatas: maksimali ar minimali matricos koordinatės reikšmė.
 **/
 unsigned int GetDebugDim(const DebugTypes &type,
                          const DataTypes &dtype);

/**
 * Matricos elemto gavimas gavimas.
 *
 * Parametrai: type - kurios matricos elementą norima sužinoti;
 *             x    - x matricos koordinatė;
 *             y    - y matricos koordinatė.
 * Rezultatas: matricos elementas.
 **/
 string GetDebugValue(const DebugTypes &type,
                      const unsigned int &x,
                      const unsigned int &y);

/**
 * Nustato iki kokio lygio reikia spresti Gauso matricą.
 *
 * Parametras: level - lygys (-1 - jokių paribojimų, 0 - tik pilna matrinca,
```

```
     *                            1 - sutvarkytas pirmas stulpelis i t.t).
     **/
     void SetGaususLevel(const int &level);

private:
    /**
     * Randa vienos lygties sprendinį.
     **/
     void SolveOne();

    /**
     * Randa lygčių sistemos sprendinį.
     **/
     void SolveMany();

    /**
     * Sprendžia lygčių sistemą Gauso metodu.
     *
     * Matrica turi būti užpildita mGausus masyve. mRadicalCount nurodo kiek
     * yra nežinomųjų, o mEquationCount - kiek lygčių. Išpradžiu yra bandoma
     * spręsti skaitiškai ir jei nepavyksta spręsti simboliškai matricą Gauso
     * metodu.
     *
     * Rezultatas: true - jei pavyko išspręsti lygčių sistemą, falese - jei ne.
     **/
     bool SolveGausus();

    /**
     * Suprastina duotą reiškinį.
     *
     * Parametras: expression - reiškinys.
     * Rezultatas: suprastintas reiškinys.
     **/
     string SimplifyExpression(const string &expression);

    /**
     * Suranda koeficientą prie nežinomojo.
     *
     * Parametrai: x - nežinomasis;
     *             element - narys, iš kurio reikia ištraukti koeficientą.
     * Rezultatas: x nežinomojo koeficientas. Jeigu ement neturėjo x nežinomojo,
     *             tai rezultatas yra nulis (0).
     **/
     string FindCoefficient(const string &x, const string &element);

    /**
     * Pateikia atsakymą naudojant yacasdll.ddl PrettyForm() funkciją.
     *
     * Gražios formos reiškinys išsaugomas mAnswer masyve.
     *
     * Parametras: expression - reiškinys.
     **/
     void UsePrettyForm(const string &expression);

    /**
     * Išmeta nereikalingus simbolius iš lygties eilutės.
     * Parametrai:
     *    equation - lygtis kurią reikia pertvarkyti.
     *    purpose  - pertvarkymo paskirtis:
     *                  2 - palieka viską išskyrus skaitmenis (0...9), lotiniškas
     *                      raides (a...z,A...Z), pliuso (+), minuso (-),
     *                      daugybos (*), dalybos (/), lygybės (=) ženklus,
     *                      laužtinius skliaustus([]), kablelį (,), tašką (.)
     *                      taip pat skaliaustus ir dar (^)
     *  bet koks skaičius - palieka viską išskyrus skaitmenis (0...9), lotiniškas
     *                      raides (a...z,A...Z) ir kablelį (,).
     * Rezultatas: sutvarkyta lygties eilutė.
     **/
     string EquationTrim(const string &equation, const int &purpose = 1) const;

    /**
     * Patikrina ar lygtis yra tvarkinga.
     *
     * Lygtis yra tvarkinga, jeigu:
     * 1. Eilutės ilgis nemažesnis kaip 3 simboliai;
     * 2. Yra viena lygybės (=) simbolis;
     * 3. Lygybės (=) simbolis yra ne pirmas ir ne paskutinis lygties simbolis;
     * 4. Pirmas lygties simbolis nėra daugybos (*) ar dalybos (/) simbolis;
     * 5. Paskutinis lygties simbolis nėra operacijos simbolis (*,/,-,+,.).
     *
```

```
    * Parametrai: equation - lygties eilutė.
    * Rezultatas: false arba true, jei atitinkamai lygties eilutė
    *                               yra klaidinga arba teisinga.
    **/
    bool CheckEquation(const string &equation) const;

private:
    typedef map<int, string> intType;
    map<int, intType> mEquations; // Duomenų masyvas saugoti lygčių informacijai.
    map<int, intType> mGausus;    // Duomenų masyvas saugoti Gauso matricai.
    map<int, string>  mRadicals;  // Ieškomi nežinomieji.

    int mMaxE;  // Maksimali mEquations masyvo x koordinatė.

    map<int, string> mAnswer; // Atsakymo eilutės.
    int mAnswerCount;         // Kiek atsakymas turi eilučių.

    int mGaususLevel;   // Iki kokio lygio reikai spręsti Gauso matricą.

    bool mPrettyForm;   // Ar atsakymą pateikti gražioje formoje.

    string mFileName;   // Log'o failo vardas.

    SimplifyLibraries mSimplify;  // Kokia yra naudojama prastinimo biblioteka.

    unsigned int mRadicalCount,  // Nežinomojų skaičius.
                 mEquationCount; // Sistemoje esančių lygčių skaičius.

    ErrorCodes mLastError;     // Paskutinės operacijos klaidos kodas.
};

#endif // SIMBOLICEQUATIONS_H
```

#---------------------- SimbolicEquations.cpp ----------------------------

```
#pragma hdrstop

#include <SysUtils.hpp>

#include <stdio.h>

#include "math_dll.h"   // naudojame simbolic.dll
#include "cyacas.h"     // naudojame yacasdll.dll
#include "SimbolicEquations.h"


/* Klasės destruktorius. */
SimbolicEquations::~SimbolicEquations()
{
}


/* Klasės konstruktorius. */
SimbolicEquations::SimbolicEquations()
{
    mEquationCount = 0;
    mLastError     = NONE;
    mSimplify      = SIMBOLIC_DLL;
    mPrettyForm    = false;
    mFileName      = "log.txt";
    mGaususLevel   = -1;
}


/* Išvalo lygčių buferį. */
void SimbolicEquations::Clear()
{
    mEquationCount = 0;

    map<int, intType>::iterator eqIt = mEquations.begin();
    while (eqIt != mEquations.end()) {
        (*eqIt).second.clear();
        eqIt++;
    }
    mEquations.clear();

    mMaxE = 0;
```

```
    map<int, intType>::iterator gausIt = mGausus.begin();
    while (gausIt != mGausus.end()) {
        (*gausIt).second.clear();
        gausIt++;
    }
    mGausus.clear();

    /* Sukuria ir išvalo log'ų failą. */
    fclose(fopen(mFileName.c_str(), "w"));

    mLastError = NONE;
}


/* Prideda lygtį į lygčių buferį. */
void SimbolicEquations::Add(const string &equation)
{
    string tmp = EquationTrim(equation);
    if (CheckEquation(tmp)) {

        mEquationCount++;

        mEquations[0][mEquationCount] = tmp;
        tmp += "=";
        int n = 2,
            m = 0;
        bool invert     = false,
             skliaustai = false;

       /* Sukame ciklą per visus lygties simbolius, ieškom narių išskyrimo
        * simbolių(+,-,=,(,) ir iškirpę tuos narius pasidedame į mEquations. */
        for (unsigned int j = 0; j < tmp.size(); j++) {
            if ((tmp[j] == '+' ||
                 tmp[j] == '-' ||
                 tmp[j] == '=' ||
                 tmp[j] == '(' ||
                 tmp[j] == ')' ||
                 j == tmp.size() - 1) &&
                 (j != 0)) {

                if (tmp[j] == '(' && !skliaustai) {
                    skliaustai = true;
                }

                if (!skliaustai) {
                   /* Tikriname ar prieš narį yra koks nors matamatinės
                    * operacijos ženklas. */
                    if (tmp[m] != '+' &&
                        tmp[m] != '-') {
                        tmp.insert(m, "+"); // Tada įterpiame pliusą.
                        j++;
                    }

                    /* Tikriname ar reikia keisti ženklą. */
                    if (invert) {
                        /* Jeigu pliusas (+). */
                        if (tmp[m] == '+') {
                            /* Vietoj jo bus minusas (-). */
                            tmp.erase(m, 1);
                            tmp.insert(m, "-");
                        } else {
                            /* Jeigu minusas (-). */
                            if (tmp[m] == '-') {
                                /* Vietoj jo bus pliusas (+). */
                                tmp.erase(m, 1);
                                tmp.insert(m, "+");
                            }
                        }
                    }

                    /* Galiausiai pasiimame vieną narį. */
                    mEquations[n][mEquationCount] = SimplifyExpression(tmp.substr(m, j - m));

                    /* Tikriname ar tik neradom lygybės ženklo. */
                    if (j != tmp.size() && tmp[j] == '=') {
                        invert = true;        // Toliau visi turės kitokį ženklą.
                        tmp.erase(j, 1); // Pašaliname = ženklą.
                    }
```

```
                    m = j;
                    n++;

                    /* Išsaugom kiek lygtyje yra narių. */
                    mEquations[1][mEquationCount] = string(IntToStr(n - 2).c_str());

                    if (mMaxE < n ) {
                        mMaxE = n;
                    }
                }

                if (tmp[j] == ')' && skliaustai) {
                    skliaustai = false;
                }
            }
        }
    } else {
        mLastError = BAD_EQUATION_FORMAT;
    }
}


/* Atlieka lygties, lygčių sistemos sprendimą. */
void SimbolicEquations::Execute(const string &radicals)
{
    unsigned int i, j, p, n;

    int tmpPos;
    string answer = "", tmpLine,ln;

    p = 0;
    n = 1;
    tmpLine = EquationTrim(radicals, 2);

    mRadicalCount = 0;

    /* Išsirenkame nežinomuosius pagal kuriuos reikia spręsti. */
    for (i = 0; i < tmpLine.size(); i++) {
        /* Jeigu kablelis, tai išsikerpam nežinomąjį. */
        if (tmpLine[i] == ',') {
            mRadicals[n] = tmpLine.substr(p, i - p);
            n++;
            p = i + 1;
        }
    }
    mRadicals[n]  = tmpLine.substr(p, i - p);
    mRadicalCount = n;

    /* Pasitikriname ar lygčių yra pakankamai. */
    if (mRadicalCount > mEquationCount) {
        /* Per mažai lygčių. */
        mLastError = NOT_ENOUGH_EQUATION;
        return;
    }

    /* Išvalome atakymų eilutes. */
    map<int, string>::iterator it = mAnswer.begin();
    while (it != mAnswer.end()) {
        (*it).second.clear();
        it++;
    }
    mAnswer.clear();
    mAnswerCount = 0;


    /* Pasitikriname kiek lygčių yra sistemoje. */
    if (mEquationCount == 1) {
        /* Reikia išspręsti paprastą lygtį. */
        SolveOne();
    } else {
        /* Daugiau negu viena lygtis -- turime lygčių sistemą */

        /* Išdėliojame lygtis normalia tvarka. */
        unsigned int a, b, j, e;
        string tmp;
        for (unsigned int i = 1; i <= mEquationCount / 2; i++) {
            j = mEquationCount - i + 1;
            a = StrToInt(mEquations[1][i].c_str());
            b = StrToInt(mEquations[1][j].c_str());
            e = (a > b)? a : b;
```

```
            for (unsigned int k = 0; k <= e + 1; k++) {
                tmp = mEquations[k][i];
                mEquations[k][i] = mEquations[k][j];
                mEquations[k][j] = tmp;
            }
        }

        /* Iškviečiame, kad išsprestų lygčių sistemą. */
        SolveMany();
    }
}


/* Grąžina rezultato eilučių skaičių. */
int SimbolicEquations::AnswerCount() const
{
    return mAnswerCount;
}


/* Grąžina rastos šaknies rezultatą. */
string SimbolicEquations::Answer(const int &answerNm)
{
    return mAnswer[answerNm];
}


/* Grąžina paskutinės atliktos operacijos klaidos kodą. */
int SimbolicEquations::Error() const
{
    return mLastError;
}


/* Nustato ar reikia pateikti ataskymą gražia forma. */
void SimbolicEquations::SetUsePrettyForm(const bool &yes)
{
    mPrettyForm = yes;
}


/* Nustato kokia prastininmo bilioteką reikia nudoti. */
void SimbolicEquations::SetSimplifyMode(const SimplifyLibraries &mode)
{
    if (mode == SIMBOLIAI_DLL ||
        mode == SIMBOLIC_DLL ||
        mode == YACASDLL_DLL ||
        mode == YACASDLL_DLL_SIMPLIFY) {
        mSimplify = mode;
    }
}


/* Nustato kur ir kokiu pavadinimu reikia išsaugoti log'ą. */
void SimbolicEquations::SetLogFileName(const string &fname)
{
    mFileName = fname;
}


/* Matricos koordinačių intervalo gavimas. */
unsigned int SimbolicEquations::GetDebugDim(const DebugTypes &type,
                                            const DataTypes &dtype)
{
    switch (type) {
        case EQUATIONS:
            switch (dtype) {
                case MIN_X:
                    return 0;
                case MIN_Y:
                    return 1;
                case MAX_X:
                    return mMaxE;
                case MAX_Y:
                    return mEquationCount;
                default:
                    return -1;
            }
        case GAUSUS_MATRIX:
            switch (dtype) {
```

```
                case MIN_X:
                    return 1;
                case MIN_Y:
                    return 0;
                case MAX_X:
                    return mRadicalCount + 1;
                case MAX_Y:
                    return mEquationCount;
                default:
                    return -1;
            }
    }
    return -1;
}


/* Matricos elemto gavimas gavimas. */
string SimbolicEquations::GetDebugValue(const DebugTypes &type,
                                        const unsigned int &x,
                                        const unsigned int &y)
{
    switch (type) {
        case EQUATIONS:
            return mEquations[x][y];
        case GAUSUS_MATRIX:
            return mGausus[x][y];
    }
    return "";
}


/* Nustato iki kokio lygio reikia spresti Gauso matricą. */
void SimbolicEquations::SetGaususLevel(const int &level)
{
    mGaususLevel = level;
}


/* Randa vienos lygties sprendinį. */
void SimbolicEquations::SolveOne()
{
    string x = mRadicals[1],
           xArg = "0",
           left = "0",
           tmp;

   /* Einame per lygties narius ir surenkame laisvuosius ir
    * prie nežinomojo esančius koeficientus. */
    int cnt = StrToInt(mEquations[1][1].c_str());
    for (int i = 2; i <= cnt + 1; i++) {
        tmp = FindCoefficient(x, mEquations[i][1]);
        if (tmp == "0") {
            /* Laisvasis narys. */
            left = SimplifyExpression("(" +
                                      left +
                                      ") + (" +
                                      mEquations[i][1] +
                                      ")");
        } else {
            /* Koeficientas prie nežinomojo. */
            xArg = SimplifyExpression("(" +
                                      xArg +
                                      ") + (" +
                                      tmp +
                                      ")");
        }
    }

    /* Tikriname ar buvo nežinomasis. */
    if (xArg == "") {
        mLastError = UNKNOWN_NOT_FOUND;
        return;
    }

    /* Surandame atsakymą. */
    tmp = SimplifyExpression("-(" + left + ")/(" + xArg + ")");

    /* Dabar galime pateikti sprendimą. */
    if (mPrettyForm) {
        /* Atsakymą reikia pateikti gražia forma. */
```

```
        mAnswerCount++;
        mAnswer[mAnswerCount] = x + "=";
        UsePrettyForm(tmp);
    } else {
        /* Atsakymą reikia pateikti paprasta forma. */
        mAnswerCount++;
        mAnswer[mAnswerCount] = x + "=" + tmp;
    }

    mLastError = NONE;
}


/* Randa lygčių sistemos sprendinį. */
void SimbolicEquations::SolveMany()
{
    unsigned int i, j, k;
    map<string, int> radId;

    /* Susidarome nežinomųjų indeksų masyvą. */
    for (i = 1; i <= mRadicalCount; i++) {
        radId[mRadicals[i]] = i;
    }

    /* Tikriname, ar visi nežinomieji yra skirtingi. */
    if (radId.size() != mRadicalCount) {
        mLastError = DUBLICATED_UNKNOWNS;
        return;
    }

    /* Išsivalome gauso matricą. */
    for (j = 1; j <= mEquationCount; j++) {
        for (i = 1; i <= mRadicalCount + 1; i++) {
            mGausus[i][j] = "0";
        }
    }
    for (i = 1; i <= mRadicalCount; i++) {
            mGausus[i][0] = mRadicals[i];
    }

    /* Susirenkame koeficientus prie nežinomųjų ir laisvuosius narius. */
    unsigned int cnt;
    bool found;
    string tmp;

    for (j = 1; j <= mEquationCount; j++) {
        cnt = StrToInt(mEquations[1][j].c_str());
        for (i = 2; i <= cnt + 1; i++) {
            found = false;
            for (k = 1; k <= mRadicalCount; k++) {
                tmp = FindCoefficient(mRadicals[k], mEquations[i][j]);
                if (tmp != "0") {
                    /* Radom koeficientą prie nežinomojo. */
                    mGausus[radId[mRadicals[k]]][j] =
                                        SimplifyExpression("(" +
                                                           mGausus[radId[mRadicals[k]]][j] +
                                                           ") + (" +
                                                           tmp +
                                                           ")");
                    found = true;
                    break;
                }
            }
            if (!found) {
                /* Laisvasis narys. */
                mGausus[mRadicalCount + 1][j] =
                                    SimplifyExpression("(" +
                                                       mGausus[mRadicalCount + 1][j] +
                                                       ") - (" +
                                                       mEquations[i][j] +
                                                       ")");
            }
        }
    }

    /* Tikriname ar yra visi lygtyse nežinomieji. */
    for (i = 1; i <= mRadicalCount + 1; i++) {
        found = false;
        for (j = 1; j <= mEquationCount; j++) {
            if (mGausus[i][j] != "0") {
```

```
                found = true;
            }
        }
        if (!found) {
            mLastError = UNKNOWN_NOT_FOUND_M;
            return;
        }
    }


    /* Tikrianame ar reikia presti Gauso matricą. */
    if (mGaususLevel == 0) {
        mLastError = NONE;
        return;
    }

    /* Dabar galime spręsti gauso matricą. */
    if (!SolveGausus()) {
        /* Lygčių sistema neturi sprendinių. */
        mLastError = SYSTEM_DO_NOT_HAVE_ROOTS;
        return;
    }

    /* Formuojame sprendinius iš Gauso matricos. */
    map<int, string> varValues;
    string answer;

    for (j = mRadicalCount; j > 0; j--) {
        answer = "";
        for (i = mRadicalCount; i >= j; i--) {
            if (i == j) {
                if (answer.size() > 1) {
                    answer += "+(" + mGausus[mRadicalCount + 1][j] + ")";
                } else {
                    answer += mGausus[mRadicalCount + 1][j];
                }
            } else {
                answer += "-(" +
                          SimplifyExpression(mGausus[i][j] +
                                             "*(" +
                                             varValues[i] +
                                             ")") +
                           ")";
            }
        }
        varValues[j] = SimplifyExpression(answer);
    }

    /* Dabar galime pateikti sprendimą. */
    if (mPrettyForm) {
        /* Atsakymą reikia pateikti gražia forma. */
        for (j = 1; j <= mRadicalCount; j++) {
            mAnswerCount++;
            mAnswer[mAnswerCount] = mRadicals[j] + "=";
            UsePrettyForm(varValues[j]);
            mAnswerCount++;
            mAnswer[mAnswerCount] = " ";
        }
    } else {
        /* Atsakymą reikia pateikti paprasta forma. */
        for (j = 1; j <= mRadicalCount; j++) {
            mAnswerCount++;
            mAnswer[mAnswerCount] = mRadicals[j] + "=" + varValues[j];
        }
    }

    mLastError = NONE;
}

/* Sprendžia lygčių sistemą Gauso metodu. */
bool SimbolicEquations::SolveGausus()
{
    unsigned int i, j, k;
    typedef map<int, long double> doubleArray;

    string      tmpLine;    // Laikina tekstinė eilutė.
    long double            tmpDouble;  // Laikinas realusis kaičius.

    map<int, doubleArray> doubleMatrix; // Skaitinė matrica.
```

```
/* Bandome skaičiuoti skaitiškai. */
try {
    /* Iš tekstinės matricos pasidarome skaitinę matricą. */
          for (j = 1; j <= mEquationCount; j++) {
          for (i = 1; i <= mRadicalCount + 1; i++) {
          doubleMatrix[i][j] = StrToFloat(mGausus[i][j].c_str());
        }
                        }

    /* Sprendžiame (skaičiuojame) matricą. */
    for (j = 1; j <= mEquationCount; j++) {
                        /* Tikriname ar koeficientas prie nežinomojo yra 0. */
                  if ((j <= mRadicalCount) &&
              (doubleMatrix[j][j] == 0)) {
                  /* Jei taip, tai reikia surasti lygtį, su kuria galima
             * būtų sukeisti. Nurodome kad negalima išspręsti
             * lygčių sistemos (galime ir nerasti su kuom sukeisti.) */
             bool solvable = false;

          for (k = j + 1; k <= mEquationCount; k++) {
                        if (doubleMatrix[j][k] != 0) {
                        for (i = j; i <= mEquationCount; i++) {
                                  tmpDouble = doubleMatrix[i][j];
                                 doubleMatrix[i][j] = doubleMatrix[i][k];
                           doubleMatrix[i][k] = tmpDouble;

                           /* Atvaizduojame pakeitimus. */
                        tmpLine = mGausus[i][j];
                                  mGausus[i][j] = mGausus[i][k];
                             mGausus[i][k] = tmpLine;
                                }
                     /* Kadangi radome su kuom sukeisti, tai galima
                      * galima išspręsti lygčių sistemą. */
                      solvable = true;
                           break;
                  }
           }
            /* Jeigu neišsprendžiama lygčių sistema, tai baigiame
             * skaičiavimus. */
             if (!solvable) {
                 return false;
             }
                   }
                   long double ff;

         /* Tikriname ar ne perteklinė lygtis. */
         if (j <= mRadicalCount) {
             ff = doubleMatrix[j][j];
         } else {
             ff = doubleMatrix[mRadicalCount][j];
         }

        /* Dėl viso pikto pasitikriname ar koeficientas prie nežinomejo
         * nėra 0 (nulis), jei taip tai nustatome kad negalima išspręsti
         * ir baigiame skaičiavimus. */
         if (!ff) {
             return false;
         }

                        /* Tikriname ar koeficientas prie nežinomojo nėra 1. */
              if (ff != 1) {
                /* Tada reikia padalinti visus eilutės narius iš to koeficiento. */
           for (i = (j <= mRadicalCount)?j:mRadicalCount; i <= mRadicalCount + 1; i++) {
                        doubleMatrix[i][j]  = doubleMatrix[i][j]/ff;
                  /* Atvaizduojame pakeitimus. */
                           mGausus[i][j] = FloatToStr(doubleMatrix[i][j]).c_str();
              }
                   }
              /* Skaičiuojame žemesnes eilutes. */
          for (k = j + 1; (k <= mEquationCount) && (j < mRadicalCount) ; k++) {
          long double aa = doubleMatrix[j][k];
                      for (i = j; i <= mRadicalCount + 1; i++) {
                      doubleMatrix[i][k] = doubleMatrix[i][k]-doubleMatrix[i][j]*aa;
                  /* Atvaizduojame pakeitimus. */
                    mGausus[i][k] = FloatToStr(doubleMatrix[i][k]).c_str();
          }
                   }

         /* Tikriname iki kokio lygio nori atlikti skaičiavimus. */
         if ((mGaususLevel != -1) && (mGaususLevel == j)) {
```

```
            return false;
        }
    }
}
/* Jei nepavyko išspręsti skaitiškai, sprendžiame simboliškai. */
catch ( ... ) {
    /* Sprendžiame (skaičiuojame) matricą. */
             for (j = 1; j <= mEquationCount; j++) {
            /* Tikriname ar koeficientas prie nežinomojo yra 0. */
         if ((j <= mRadicalCount) &&
            (mGausus[j][j] == "0")) {
                    /* Jei taip, tai reikia surasti lygtį, su kuria galima
                 * būtų sukeisti. Nurodome kad negalima išspręsti
            * lygčių sistemos (galime ir nerasti su kuom sukeisti.) */
            bool solvable = false;

              for (k = j + 1; k <= mEquationCount; k++) {
              if (mGausus[j][k] != "0") {
                           for (i = j; i <= mEquationCount; i++) {
                                  tmpLine = mGausus[i][j];
                              mGausus[i][j] = mGausus[i][k];
                          mGausus[i][k] = tmpLine;
                    }
                       /* Kadangi radome su kuom sukeisti, tai galima
                       * galima išspręsti lygčių sistemą. */
                       solvable = true;
                              break;
                             }
                    }
              /* Jeigu neišsprendžiama lygčių sistema, tai baigiame
               * skaičiavimus. */
               if (!solvable) {
                   return false;
               }
                     }
          string f;

         /* Tikriname ar ne perteklinė lygtis. */
         if (j <= mRadicalCount) {
             f = SimplifyExpression(mGausus[j][j]);
         } else {
             f = SimplifyExpression(mGausus[mRadicalCount][j]);
         }

        /* Dėl viso pikto pasitikriname ar koeficientas prie nežinomojo
         * nėra 0 (nulis), jei taip tai nustatome kad negalima išspręsti
         * ir baigiame skaičiavimus. */
         if (f == "0") {
             return false;
         }

         /* Tikriname ar koeficientas prie nežinomojo nėra 1. */
                   if (f != "1") {
                   /* Tada reikia padalinti visus eilutės narius iš to koeficiento. */
                 for (i = (j <= mRadicalCount)?j:mRadicalCount; i <= mRadicalCount + 1; i++) {
                   /* Bandome padalinti skaitiškai. */
                          try {
                                                  double a = StrToFloat(mGausus[i][j].c_str());
                          double b = StrToFloat(f.c_str());
                                 mGausus[i][j] = FloatToStr(float(a/b)).c_str();
                        }
                   /* Jei negalima skaitiškai, teks naudoti simbolinę dalybą. */
                                                                catch ( ... ) {
                                                                                mGausus[i][j] =
                               SimplifyExpression("(" +
                                                  SimplifyExpression(mGausus[i][j]) +
                                                  ")/(" +
                                                  f +
                                                  ")");
                          }
                         }
              }
                   /* Skaičiuojame žemesnes eilutes. */
                   for (k = j + 1; k <= mRadicalCount; k++) {
                   string a = SimplifyExpression(mGausus[j][k]);
                           for (i = j; i <= mRadicalCount + 1; i++) {
                     /* Bandome sksičiuoti skaitiškai. */
                          try {
                                                  double a1 = StrToFloat(a.c_str());
                       double b1 = StrToFloat(mGausus[i][j].c_str());
```

```
                                double c1 = StrToFloat(mGausus[i][k].c_str());
                                  mGausus[i][k] = FloatToStr(c1 - b1 * a1).c_str();
                            }
                        /* Jei nepavyko - simboliškai. */
                                                                    catch ( ... ) {
                                    string b = SimplifyExpression(mGausus[i][j].c_str());
                                string c = SimplifyExpression(mGausus[i][k].c_str());
                             mGausus[i][k] = SimplifyExpression(c +
                                                                "-(" +
                                                                b +
                                                                ")*(" +
                                                                a +
                                                                ")");

                            }
                    }
                        }

                /* Tikriname iki kokio lygio nori atlikti skaičiavimus. */
                if ((mGaususLevel != -1) && (mGaususLevel == j)) {
                    return false;
                }
                     }
        }
        /* Tikriname jei buvo perteklinės lygtys ar yra sprendinys. */
        if (mRadicalCount != mEquationCount) {
            for (j = mRadicalCount; j <= mEquationCount; j++) {
                if (mGausus[mRadicalCount + 1][mRadicalCount] !=
                    mGausus[mRadicalCount + 1][j]) {
                    return false;
                }
            }
        }

        return true;
    }


    /* Suprastina duotą reiškinį. */
    string SimbolicEquations::SimplifyExpression(const string &expression)
    {
        typedef void (*func_simb)(char *);
        string corectVall,
               modVal,
               data;
        char *val;
        FILE *file;

        corectVall = expression;

        /* Simboliai.dll apėjimas. */
        if (mSimplify == SIMBOLIAI_DLL) {
            unsigned int i = 0;
            bool can;

            if (corectVall[0] == '+') {
                corectVall.erase(0, 1);
            }

            while (i < corectVall.size()) {
                can = false;
                while (corectVall[i] == '(' && i < corectVall.size()) {
                    i++;
                    can = true;
                }

                if (can && corectVall[i] == '+') {
                    corectVall.erase(i, 1);
                }

                i++;
            }
        }

        /* Ieškome kablelio, jei radome, pakeičiame tašku. */
        for (unsigned int i = 1; i < corectVall.size(); i++) {
            if (corectVall[i] == ',') {
                corectVall.erase(i, 1);
                corectVall.insert(i, ".");
            }
```

```
    }

    func_simb simboliai;
    HINSTANCE simb_load=LoadLibrary("Simboliai.dll");
    simboliai = (func_simb)GetProcAddress(simb_load, "_simboliai");
    val = new char[2*corectVall.size() + 2];
    strcpy(val, corectVall.c_str());

    /* Atidarome logo failą ir išsaugome tai ka norime suprastinti. */
    file = fopen(mFileName.c_str(), "a+");
    data = "before: " + corectVall + "\n";
    fprintf(file, data.c_str());
    fclose(file);

    /* Bandome prastinti. */
    switch (mSimplify)
    {
      case SIMBOLIAI_DLL:
            simboliai(val);
            modVal = val;
            break;
      case SIMBOLIC_DLL:
            symbolic_calculate(val, val, 2*corectVall.size() + 2);
            modVal = val;
            break;

      case YACASDLL_DLL:
      case YACASDLL_DLL_SIMPLIFY:
            yacas_init();
            yacas_eval(val);
            if (mSimplify == YACASDLL_DLL_SIMPLIFY) yacas_eval("Simplify(%)");
            if (yacas_error()) {
                file = fopen(mFileName.c_str(), "a+");
                data = "yacas_error: ";
                fprintf(file, data.c_str());
                fprintf(file, yacas_error());
                data = "\n\n";
                fprintf(file, data.c_str());
                fclose(file);
            } else {
                modVal = yacas_result();
                modVal.resize(modVal.size() - 1);
            }
            yacas_exit();
            break;
      default:
            modVal = "0";
    }

    delete val;

    /* Jeigu prastinimas nepakibo išsaugome ką suprastino į log failą. */
    file = fopen(mFileName.c_str(), "a+");
    data = "after : " + modVal + "\n\n";
    fprintf(file, data.c_str());
    fclose(file);

    return modVal;
}


/* Suranda koeficientą prie nežinomojo. */
string SimbolicEquations::FindCoefficient(const string &x,
                                          const string &element)
{
    string tmpLine = element,
           answer  = "";
    int tmpPos;
    unsigned int pos = 0;

    if (x != element) {
        tmpPos = tmpLine.find(x);
        if (tmpPos >= 0) {
            pos  += tmpPos;
            /* Reikia įsitikinti, kad radome nežinomąjį. */
            if ((tmpPos >= 1) &&
                (tmpLine[tmpPos - 1] != '*') &&
                (tmpLine[tmpPos - 1] != '/') &&
                (tmpLine[tmpPos - 1] != '+') &&
                (tmpLine[tmpPos - 1] != '-') &&
```

```
                (tmpLine[tmpPos - 1] != '(')) {
                return "0";
            }

            tmpPos += x.size();
            if ((tmpPos < tmpLine.size()) &&
                (tmpLine[tmpPos] != '*') &&
                (tmpLine[tmpPos] != '/') &&
                (tmpLine[tmpPos] != '+') &&
                (tmpLine[tmpPos] != '-') &&
                (tmpLine[tmpPos] != ')')) {
                return "0";
            }
        } else {
            return "0";
        }

        answer = element.substr(0, pos);
        answer += "1";
 //       answer = element.substr(0, pos);
/*      if (answer.size() == 0) {
          answer += "1";
        }

        if (answer[answer.size() - 1 ] == '*') {
            answer.erase(answer.size() - 1, 1);
        }

        if (answer[answer.size() - 1 ] == '/') {
            answer.erase(answer.size() - 1, 1);
            answer = "(1/" + answer +")";
        }
*/
        answer += element.substr(pos + x.size(), element.size() - pos + x.size());

        answer = SimplifyExpression(answer);
        return answer;
    } else {
        return "1";
    }
}


/* Pateikia atsakymą naudojant yacasdll.ddl PrettyForm() funkciją. */
void SimbolicEquations::UsePrettyForm(const string &expression)
{
    string corectVall,
           modVal,
           data,
           yacasResult;

    char *val;

    corectVall = expression;

    /* Ieškome kablelio, jei radome, pakeičiame tašku. */
    for (unsigned int i = 1; i < corectVall.size(); i++) {
        if (corectVall[i] == ',') {
            corectVall.erase(i, 1);
            corectVall.insert(i, ".");
        }
    }

    val = new char[2*corectVall.size() + 2];
    strcpy(val, corectVall.c_str());

    yacas_init();
    yacas_eval(val);
    yacas_eval("Expand(%,sc2)");
    yacas_eval("PrettyForm(%)");
    if (!yacas_error()) {
        yacasResult = string(yacas_output());
        int pos;
        while ((pos = yacasResult.find(string("\n\n"))) > 0) {
            yacasResult[pos + 1] = '?';
        }
        map<int, string> lines;

        int cnt = yacasResult.size(),
            p   = 1,
```

```
            j   = 0;
        for (int i = 1; i < cnt; i++) {
            if (yacasResult[i] == '\n' && p != i) {
                j++;
                lines[j] = lines[j] + yacasResult.substr(p , i - p);
                p = i + 1;
            } else {
                if (yacasResult[i] == '?') {
                    j = 0;
                    p = i + 1;
                }
            }
        }

        cnt = lines.size();
        for (int i = 1; i <= cnt; i++) {
            mAnswerCount++;
            mAnswer[mAnswerCount] = lines[i];
        }
    } else {
        mAnswerCount++;
        mAnswer[mAnswerCount] = string(yacas_error());
    }
    delete val;
}


/* Išmeta nereikalingus simbolius iš lygties eilutės. */
string SimbolicEquations::EquationTrim(const string &equation,
                                       const int &purpose) const
{
    string tmp = equation;
    unsigned int i = 0;

    if (purpose == 2) {
        while (i < tmp.size()) {
            if ((tmp[i]>='0') && (tmp[i]<='9') ||
                (tmp[i]>='a') && (tmp[i]<='z') ||
                (tmp[i]>='A') && (tmp[i]<='Z') ||
                (tmp[i]==',')) {
                i++;
            } else {
                tmp.erase(i, 1);
            }
        }
    } else {
        while (i < tmp.size()) {
            if ((tmp[i]>='0') && (tmp[i]<='9') ||
                (tmp[i]>='a') && (tmp[i]<='z') ||
                (tmp[i]>='A') && (tmp[i]<='Z') ||
                (tmp[i]=='+') || (tmp[i]=='-') ||
                (tmp[i]=='*') || (tmp[i]=='/') ||
                (tmp[i]=='(') || (tmp[i]==')') ||
                (tmp[i]=='=') || (tmp[i]=='.')) {
                i++;
            } else {
                tmp.erase(i, 1);
            }
        }
    }

    return tmp;
}


/* Patikrina ar lygtis yra tvarkinga. */
bool SimbolicEquations::CheckEquation(const string &equation) const
{
    /* Tikriname 2 sąlyga. */
    int i = 0;
    unsigned int last = equation.length() - 1;

    for (unsigned j = 0; j <= last; j++) {
        if (equation[j] == '=') {
            i++;
        }
    }
    bool s2 = i == 1;

    /* Tikriname 1, 3 - 5 sąlygas. */
```

```
    bool s1 = equation.size() >= 3,
         s3 = equation[0] != '=' &&
              equation[last] != '=',
         s4 = equation[0] != '*' &&
              equation[0] != '/',
         s5 = equation[last] != '*' &&
              equation[last] != '/' &&
              equation[last] != '+' &&
              equation[last] != '-' &&
              equation[last] != '.';

    return s1 && s2 && s3 && s4 && s5;
}


#pragma package(smart_init)
```

#---------------------------- SimbolicMatrixes.h -----------------------------

```
#ifndef SIMBOLICMATRIXES_H
#define SIMBOLICMATRIXES_H

#include <string>
#include <map>

using namespace std;

/**
 * Simbolinių matricų skaičiavimo klasė.
 *
 * Versija: 0.39
 **/
class SimbolicMatrixes
{
public:
    // Galimų klaidų sąrašas.
    enum ErrorCodes {
        NONE,                              // Jokių klaidų.
        BAD_MATRIX_FORMAT,                 // Blogas matricos formatas.
        BAD_ANSWER_NAME,                   // Blogas matricos veiksmų rezultatų pavadimas.
        UNKNOWN_OPERATION,                 // Nesuprantama operacija.
        DIMENSIONS_NOT_EQUAL,              // Matricų matmenys nevienodi.
        INTERVALS_ENDS_NOT_EQUAL,          // Matricų galų intervalai nelygus.
        INTERVALS_ENDS_NOT_FOR_MULTIPLY,   // Matricų galų intervalai netinka sandaugai.
        ELEMENTS_INDEXES_NOT_EQUAL,        // Matricų elementų indeksai nelygus.
        ELEMENTS_INDEXES_NOT_FOR_MULTIPLY, // Matricų elementų indeksai netinka sandaugai.
    };

public:
   /**
    * Klasės destruktorius.
    **/
    ~SimbolicMatrixes();

   /**
    * Klasės konstruktorius.
    *
    * Parametras: ignoreElementIndexes - nurodo ar ignoruoti elementų indeksus.
    **/
    SimbolicMatrixes(bool ignoreElementIndexes = false);

   /**
    * Išvalo matricų buferį.
    **/
    void Clear();

   /**
    * Prideda matricą į matricų buferį.
    *
    * Jei matricų buferyje jau buvo tokia matrica,
    * tai senoji yra perrašoma.
    *
    * Parametras: matrix - matrica.
    **/
    void Add(const string &matrix);

   /**
    * Įvykdo matricų veiksmus.
```

```
     *
     * Parametras: condition - ką reikia apskaičiuoti.
     **/
     void Execute(const string &condition);

    /**
     * Grąžina matricų vieksmų rezultatą.
     *
     * Rezultatas: Matricų veiksmų rezultatas.
     **/
     string Answer() const;

    /**
     * Grąžina paskutinės atliktos operacijos klaidos kodą.
     *
     * Rezultatas: Paskutinės atliktos operacijos klaidos kodas.
     **/
     int Error() const;

private:
    /**
     * Išmeta nereikalingus simbolius iš matricos eilutės.
     *
     * Palieka viską išskyrus skaitmenis (0...9), lotiniškas
     * raides (a...z,A...Z), pliuso (+), minuso (-),
     * daugybos (*), dalybos (/), lygybės (=) ženklus,
     * laužtinius skliaustus([]), kablelį (,), daugiau (>)
     * taip pat skaliaustus ir dar (^)
     *
     * Parametras: matrix - matricos eilutė.
     * Rezultatas: sutvarkyta matricos eilutė.
     **/
     string MatrixTrim(const string &matrix) const;

    /**
     * Patikrina ar matrica yra tvarkinga.
     *
     * Matrica yra tvarkinga, jei:
     * 1. yra žodelis "matrica" nuo trečio simbolio;
     * 2. antras simbolis yra lygybė "=";
     * 3. paskutinis simbolis yra uždarantys skliaustai ")";
     * 4. eilutės ilgis nemazesnis kaip 20 simboliu (min turi
     *    būti kakžkas panašaus į tai: A=matrica(m,n,a[i,j]));
     * 5. Turi turėti nemažiau 3 kablelių.
     *
     * Parametrai: matrix - matricos eilutė.
     * Rezultatai: false arba true, jei atitinkamai matricos eilutė
     *                              yra klaidinga arba teisinga.
     **/
     bool CheckMatrix(const string &matrix) const;

    /**
     * Patikrina ar egzistuoja matrica.
     *
     * Ieško ar matricos pavadinimas nėra toks pat, kaip
     * kokios nors matricos esančios data masyve.
     *
     * Parametrai: name - matricos pavadinimas.
     * Rezultatai: true, jei yra tokia matrica.
     **/
     bool ExistMatrix(const char &name) const;

    /**
     * Tikriname ar matricos tinkamos atlikti oparacijas.
     *
     * Patikrina ar matricų elementų koeficientai tvarkingi.
     *
     * Paramaetrai: value - operacijos reikšmė, pvz.: A+B, A*B, A^t
     *              type - nurodo koks veiskmas yra atliekamas:
     *                     1 - sudėtis arba atimtis;
     *                     2 - sandauga;
     *                     3 - transponavimas;
     * Rezultatai: true, jei tinkami.
     **/
     bool CheckArgs(const string &value, const int &type);

    /**
     * Mažąją raidę paverčia didžiąją.
     *
     * Parametrai: simbol - raidė, kurią reikia versti į didžiąją.
```

```
    * Rezultatai: paversta raidė į didžiąją.
    **/
    char UpCase(const char &simbol) const;

private:
    typedef map<int, string> strMatrixType;
    typedef map<int, int>    intMatrixType;
    map<char, strMatrixType> mStrData; // Duomenų masyvas saugoti matricų tekstinei informacijai.
    map<char, intMatrixType> mIntData; // Duomenų masyvas saugoti matricų skaitinei informacijai.

    ErrorCodes mLastError;        // Paskutinės operacijos klaidos kodas.

    string mAnswerMatrix;         // Operacijos atsakymas.
    bool   mIgnoreElementIndexes; // Ar ignoruoti matricų elementų indeksus.
};

#endif // SIMBOLICMATRIXES_H
```

------------------------------ SimbolicMatrixes.cpp ------------------------

```
#include "SimbolicMatrixes.h"


// Klasės destruktorius.
SimbolicMatrixes::~SimbolicMatrixes()
{
}


// Klasės konstruktorius.
SimbolicMatrixes::SimbolicMatrixes(bool ignoreElementIndexes)
:mIgnoreElementIndexes(ignoreElementIndexes)
{
    mLastError = NONE;
}


// Išvalo matricų buferį.
void SimbolicMatrixes::Clear()
{
    map<char, strMatrixType>::iterator strIt = mStrData.begin();
    while (strIt != mStrData.end()) {
        (*strIt).second.clear();
        strIt++;
    }
    mStrData.clear();

    map<char, intMatrixType>::iterator intIt = mIntData.begin();
    while (intIt != mIntData.end()) {
        (*intIt).second.clear();
        intIt++;
    }
    mIntData.clear();

    mLastError = NONE;
}


// Prideda matricą į matricų buferį.
void SimbolicMatrixes::Add(const string &matrix)
{
    string tmp = MatrixTrim(matrix);
    if (CheckMatrix(tmp)) {
        /* Bandome išrinkti matricos dalis. */
        char M = tmp[0];
        tmp.erase(0,10); // length("M=matrica(") = 10
        tmp.erase(tmp.size() - 1, 1);
        int j = 1;//,
            //m = 0;
        string t = "";
        bool all = false;
        for (unsigned int k = 0; k < tmp.size(); k++) {
            if (all) {
                t += tmp[k];
            } else {
                if (tmp[k] == ',') {
                    mStrData[M][j] = t;
                    //m = k + 1;
```

```
                    j++;
                    t = "";
                } else {
                    if (tmp[k] == '[') all = true;
                    t += tmp[k];
                }
            }
        }
        mStrData[M][j] = t;
        mIntData[M][0] = j - 1; // Kelių matmenų matrica.
        j++;
        mStrData[M][j] = t[0];
        mStrData[M][j + 1] = t[2];
        mStrData[M][j + 2] = t[4];
        if (mIntData[M][0] == 3) {
            /* Jei trimatė matrica. */
            mStrData[M][j + 3] = t[6];
            t.erase(0, 9);
            if (!t.empty()) {
                mStrData[M][j + 4] = t;
            } else {
                mStrData[M][j + 4] = "";
            }
        } else {
            /* Jei dvimatė matrica. */
            t.erase(0, 7);
            if (!t.empty()) {
                mStrData[M][j + 3] = t;
            } else {
                mStrData[M][j + 3] = "";
            }
        }
        mLastError = NONE;
    } else {
        mLastError = BAD_MATRIX_FORMAT;
    }
}


// Įvykdo matricų veiksmus.
void SimbolicMatrixes::Execute(const string &condition)
{
    string tmp = MatrixTrim(condition),
           answer = "";
    char M = tmp[0];
   /* Ieškome ar rezultato matricos pavadinimas nėra toks pat, kaip
    * kokios nors matricos. */
    if (!ExistMatrix(M)) {
        /* Ieškom kokią operaciją reikia atlikti. */
        char action;
        /* Ar skaičiuosime determinantą: |A|. */
        if (tmp[2] == '|') {
            action = '|';
        } else {
            /* Ar transponuosime: A^t
             * ar normalizuosime: A^n. */
            if (tmp[3] == '^') {
                action = tmp[4];
            } else {
                /* Ar atliksime aritmetinius veiskmus:
                 * A+B, A-B, A*B */
                action  = tmp[3];
            }
        }
        tmp.erase(0,2);
        char A = tmp[0],
             B = tmp[2];
        /* Atliksime duotąjį veiksmą.*/
        switch (action) {
            case '+': // Matricų suma.
                    if (CheckArgs(tmp, 1)) {
                        answer = UpCase(M);
                        answer += " = matrica(";
                        answer += mStrData[A][1];
                        answer += ", ";
                        answer += mStrData[A][2];
                        answer += ", ";
                        if (mIntData[A][0] == 3) {
                            // Jei trimatė matrica.
                            answer += mStrData[A][3];
```

```
                answer += ", ";
                answer += M;
                answer += "[";
                answer += mStrData[A][6];
                answer += ", ";
                answer += mStrData[A][7];
                answer += ", ";
                answer += mStrData[A][8];
                answer += "] = ";
                answer += mStrData[A][9].empty()?mStrData[A][4]:mStrData[A][9];
                answer += " + ";
                answer += mStrData[B][9].empty()?mStrData[B][4]:mStrData[B][9];
                answer += ")";
            } else {
                // Jei dvimatė matrica.
                answer += M;
                answer += "[";
                answer += mStrData[A][5];
                answer += ", ";
                answer += mStrData[A][6];
                answer += "] = ";
                answer += mStrData[A][7].empty()?mStrData[A][3]:mStrData[A][7];
                answer += " + ";
                answer += mStrData[B][7].empty()?mStrData[B][3]:mStrData[B][7];
                answer += ")";
            }
        }
    break;
case '-': // Matricų skirtumas.
        if (CheckArgs(tmp, 1)) {
            answer = UpCase(M);
            answer += " = matrica(";
            answer += mStrData[A][1];
            answer += ", ";
            answer += mStrData[A][2];
            answer += ", ";
            answer += M;
            answer += "[";
            answer += mStrData[A][5];
            answer += ", ";
            answer += mStrData[A][6];
            answer += "] = ";
            answer += mStrData[A][7].empty()?mStrData[A][3]:mStrData[A][7];
            answer += " - ";
            answer += mStrData[B][7].empty()?mStrData[B][3]:mStrData[B][7];
            answer += ")";
        }
    break;
case '*': // Matricų sandauga.
        if (CheckArgs(tmp, 2)) {
            answer = UpCase(M);
            answer += " = matrica(";
            answer += mStrData[A][1];
            answer += ", ";
            answer += mStrData[B][2];
            answer += ", ";
            answer += M;
            answer += "[";
            answer += mStrData[A][5];
            answer += ", ";
            answer += mStrData[B][6];
            answer += "] = ";
            answer += "sum({";
            answer += mStrData[A][6];
            answer += "->";
            answer += mStrData[A][2];
            answer += "}, (";
            answer += mStrData[A][7].empty()?mStrData[A][3]:mStrData[A][7];
            answer += " * ";
            answer += mStrData[B][7].empty()?mStrData[B][3]:mStrData[B][7];
            answer += ")))";
        }
    break;
case 't': // Matricų transponavimas.
        if (CheckArgs(tmp, 3)) {
            answer = UpCase(M);
            answer += " = matrica(";
            answer += mStrData[A][2];
            answer += ", ";
            answer += mStrData[A][1];
```

```
                        answer += ", ";
                        answer += M;
                        answer += "[";
                        answer += mStrData[A][6];
                        answer += ", ";
                        answer += mStrData[A][5];
                        answer += "] = ";
                        answer += mStrData[A][4];
                        answer += "[";
                        answer += mStrData[A][5];
                        answer += ", ";
                        answer += mStrData[A][6];
                        answer += "])";
                    }
                break;
            default:
                mLastError = UNKNOWN_OPERATION;
        }

    } else {
        mLastError = BAD_ANSWER_NAME;
    }
    mAnswerMatrix = answer;
}


// Grąžina matricų vieksmų rezultatą.
string SimbolicMatrixes::Answer() const
{
    return mAnswerMatrix;
}


// Grąžina paskutinės atliktos operacijos klaidos kodą.
int SimbolicMatrixes::Error() const
{
    return mLastError;
}


// Išmeta nereikalingus simbolius iš matricos eilutės.
string SimbolicMatrixes::MatrixTrim(const string &matrix) const
{
    string tmp = matrix;
    unsigned int i = 0;

    while (i < tmp.size()) {
        if ((tmp[i]>='0') && (tmp[i]<='9') ||
           (tmp[i]>='a') && (tmp[i]<='z') ||
           (tmp[i]>='A') && (tmp[i]<='Z') ||
           (tmp[i]=='+') || (tmp[i]=='-') ||
           (tmp[i]=='*') || (tmp[i]=='/') ||
           (tmp[i]=='=') || (tmp[i]=='^') ||
           (tmp[i]=='[') || (tmp[i]==']') ||
           (tmp[i]=='(') || (tmp[i]==')') ||
           (tmp[i]==',') || (tmp[i]=='>')) {
            i++;
        } else {
            tmp.erase(i, 1);
        }
    }

    return tmp;
}


// Patikrina ar matrica yra tvarkinga.
bool SimbolicMatrixes::CheckMatrix(const string &matrix) const
{
    /* Tikriname 5 sąlyga. */
    int i = 0;
    for (unsigned j = 0; j < matrix.length(); j++) {
        if (matrix[j] == ',') {
            i++;
        }
    }
    bool s5 = i >= 3;

    /* Tikriname 1 - 4 sąlygas. */
    bool s1 = matrix.find("matrica") == 2,
```

```
         s2 = matrix[1] == '=',
         s3 = matrix[matrix.size() - 1] == ')',
         s4 = matrix.size() >= 20;

    return s1 && s2 && s3 && s4 && s5;
}


// Patikrina ar egzistuoja matrica.
bool SimbolicMatrixes::ExistMatrix(const char &name) const
{
    map<char, strMatrixType>::const_iterator it = mStrData.find(name);
    return it != mStrData.end();
}


// Tikriname ar matricos tinkamos atlikti oparacijas.
bool SimbolicMatrixes::CheckArgs(const string &value, const int &type)
{
    /* Tikriname transponavimą*/
    if (type == 3) {
        return true;
    }

    bool ok;

    char A = value[0],
         B = value[2];

    /* Tikriname ar matricų matmenys sutampa. */
    if (mIntData[A][0] != mIntData[B][0]) {
        mLastError = DIMENSIONS_NOT_EQUAL;
        return false;
    }

    /* Tikriname matricų galų intervalus. */
    if (type == 1) {
        // Sudėtis arba atimtis.
        if (mIntData[A][0] == 3) {
            // Trimatė matrica.
            ok  = (mStrData[A][1] == mStrData[B][1]) &&
                  (mStrData[A][2] == mStrData[B][2]) &&
                  (mStrData[A][3] == mStrData[B][3]);
        } else {
            // Dvimatė matrica.
            ok  = (mStrData[A][1] == mStrData[B][1]) &&
                  (mStrData[A][2] == mStrData[B][2]);
        }
    } else {
        // Sandauga.
        ok  = mStrData[A][2] == mStrData[B][1];
    }

    if (!ok) {
        /* Matricų intervalų galai nesutampa. */
        if (type == 1) {
            mLastError = INTERVALS_ENDS_NOT_EQUAL;
        } else {
            mLastError = INTERVALS_ENDS_NOT_FOR_MULTIPLY;
        }
    } else {
        /* Tikriname ar reikia tikrinti matricos elementu indeksus. */
        if (!mIgnoreElementIndexes) {
            if (type == 1) {
                /* Jei suma ar skirtumas */
                if (mIntData[A][0] == 3) {
                    // Trimatė matrica.
                    ok  = (mStrData[A][6] == mStrData[B][6]) &&
                          (mStrData[A][7] == mStrData[B][7]) &&
                          (mStrData[A][8] == mStrData[B][8]);
                } else {
                    // Dvimatė matrica.
                    ok  = (mStrData[A][5] == mStrData[B][5]) &&
                          (mStrData[A][6] == mStrData[B][6]);
                }
                if (!ok) {
                    mLastError = ELEMENTS_INDEXES_NOT_EQUAL;
                }
            } else {
                /* Jei sandauga */
```

```
                ok  = mStrData[A][6] == mStrData[B][5];
                if (!ok) {
                    mLastError = ELEMENTS_INDEXES_NOT_FOR_MULTIPLY;
                }
            }
        }
    }

    return ok;
}


// Mažąją raidę paverčia didžiąją.
char SimbolicMatrixes::UpCase(const char &simbol) const
{
    if ((simbol >= 'a') && (simbol <= 'z')) {
        return simbol - 32;
    } else {
        return simbol;
    }
}
```

Priedas 3

Pateikiame žinių bazės serverio išeities kodą.

```
#------------------------------------------------------------------------------------------------------------------

CREATE SEQUENCE "list_id_seq" start 1 increment 1 maxvalue 9223372036854775807
minvalue 1 cache 1;


CREATE SEQUENCE "users_id_seq" start 1 increment 1 maxvalue 9223372036854775807
minvalue 1 cache 1;


CREATE TABLE "users" (
           "id" integer DEFAULT nextval('"users_id_seq"'::text) NOT NULL,
           "name" character varying(100) NOT NULL,
           "email" character varying(100) NOT NULL,
           Constraint "users_pkey" Primary Key ("id")
);


CREATE SEQUENCE "category_id_seq" start 1 increment 1 maxvalue
9223372036854775807 minvalue 1 cache 1;



CREATE TABLE "categories" (
           "id" integer DEFAULT nextval('"category_id_seq"'::text) NOT NULL,
           "name" character varying(100) NOT NULL,
           Constraint "categories_pkey" Primary Key ("id")
);



CREATE TABLE "types" (
           "name" character varying(1) NOT NULL,
           Constraint "types_pkey" Primary Key ("name")
);



CREATE SEQUENCE "simplifications_id_seq" start 1 increment 1 maxvalue
9223372036854775807 minvalue 1 cache 1;



CREATE TABLE "simplifications" (
           "id" integer DEFAULT nextval('"simplifications_id_seq"'::text) NOT
NULL,
           "name" character varying(100) NOT NULL,
           Constraint "simplifications_pkey" Primary Key ("id")
);



CREATE SEQUENCE "data_id_seq" start 1 increment 1 maxvalue 9223372036854775807
minvalue 1 cache 1;



CREATE TABLE "data" (
           "id" integer DEFAULT nextval('"data_id_seq"'::text) NOT NULL,
```

```
            "user_id" integer NOT NULL,
            "category_id" integer NOT NULL,
            "type" character varying(1) NOT NULL,
            "condition" text NOT NULL,
            "solution" text NOT NULL,
            "simplification" integer NOT NULL,
            "simplify" boolean DEFAULT 'f'::bool NOT NULL,
            Constraint "data_pkey" Primary Key ("id")
);



CREATE TABLE "user_categories" (
            "user_id" integer NOT NULL,
            "category_id" integer NOT NULL
);



CREATE VIEW "usr_cat_view" as SELECT users.name AS user_name, categories.name AS
category_name FROM users, categories, user_categories WHERE ((users.id =
user_categories.user_id) AND (categories.id = user_categories.category_id));



CREATE SEQUENCE "system_id_seq" start 1 increment 1 maxvalue 9223372036854775807
minvalue 1 cache 1;



CREATE TABLE "system" (
            "id" integer DEFAULT nextval('"system_id_seq"'::text) NOT NULL,
            "version" character varying(16) NOT NULL,
            Constraint "system_pkey" Primary Key ("id")
);




COPY "users" FROM stdin;
\.


COPY "categories" FROM stdin;
\.

COPY "types" FROM stdin;
M
L
\.



COPY "simplifications" FROM stdin;
1           Simboliai.dll
2           simbolic.dll
3           yacasdll.dll
4           yacasdll.dll + Simplify()
\.


COPY "data" FROM stdin;
\.
--
```

```
COPY "user_categories" FROM stdin;
\.


COPY "system" FROM stdin;
1          0.14.32.2805
\.


CREATE UNIQUE INDEX user_categories_user_id_key ON user_categories USING btree
(user_id, category_id);



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON "data"  FROM
"users" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_check_ins" ('<unnamed>', 'data', 'users', 'UNSPECIFIED', 'user_id',
'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "users"  FROM "data" NOT
DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_del" ('<unnamed>', 'data', 'users', 'UNSPECIFIED', 'user_id',
'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "users"  FROM "data" NOT
DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_upd" ('<unnamed>', 'data', 'users', 'UNSPECIFIED', 'user_id',
'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON "data"  FROM
"categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_check_ins" ('<unnamed>', 'data', 'categories', 'UNSPECIFIED',
'category_id', 'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "categories"  FROM "data"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_del" ('<unnamed>', 'data', 'categories', 'UNSPECIFIED',
'category_id', 'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "categories"  FROM "data"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_upd" ('<unnamed>', 'data', 'categories', 'UNSPECIFIED',
'category_id', 'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON "data"  FROM
"types" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_check_ins" ('<unnamed>', 'data', 'types', 'UNSPECIFIED', 'type',
'name');
```

```
CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "types"  FROM "data" NOT
DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_del" ('<unnamed>', 'data', 'types', 'UNSPECIFIED', 'type',
'name');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "types"  FROM "data" NOT
DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_upd" ('<unnamed>', 'data', 'types', 'UNSPECIFIED', 'type',
'name');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON "data"  FROM
"simplifications" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE
PROCEDURE "RI_FKey_check_ins" ('<unnamed>', 'data', 'simplifications',
'UNSPECIFIED', 'simplification', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "simplifications"  FROM
"data" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_del" ('<unnamed>', 'data', 'simplifications', 'UNSPECIFIED',
'simplification', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "simplifications"  FROM
"data" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
"RI_FKey_cascade_upd" ('<unnamed>', 'data', 'simplifications', 'UNSPECIFIED',
'simplification', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON
"user_categories"  FROM "users" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW
EXECUTE PROCEDURE "RI_FKey_check_ins" ('<unnamed>', 'user_categories', 'users',
'UNSPECIFIED', 'user_id', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "users"  FROM
"user_categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE
PROCEDURE "RI_FKey_cascade_del" ('<unnamed>', 'user_categories', 'users',
'UNSPECIFIED', 'user_id', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "users"  FROM
"user_categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE
PROCEDURE "RI_FKey_cascade_upd" ('<unnamed>', 'user_categories', 'users',
'UNSPECIFIED', 'user_id', 'id');


CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER INSERT OR UPDATE ON
"user_categories"  FROM "categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH
ROW EXECUTE PROCEDURE "RI_FKey_check_ins" ('<unnamed>', 'user_categories',
'categories', 'UNSPECIFIED', 'category_id', 'id');
```

```
CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER DELETE ON "categories"  FROM
"user_categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE
PROCEDURE "RI_FKey_cascade_del" ('<unnamed>', 'user_categories', 'categories',
'UNSPECIFIED', 'category_id', 'id');



CREATE CONSTRAINT TRIGGER "<unnamed>" AFTER UPDATE ON "categories"  FROM
"user_categories" NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE
PROCEDURE "RI_FKey_cascade_upd" ('<unnamed>', 'user_categories', 'categories',
'UNSPECIFIED', 'category_id', 'id');


SELECT setval ('"list_id_seq"', 1, false);


SELECT setval ('"users_id_seq"', 1, true);


SELECT setval ('"category_id_seq"', 1, true);


SELECT setval ('"simplifications_id_seq"', 4, true);


SELECT setval ('"data_id_seq"', 1, true);


SELECT setval ('"system_id_seq"', 2, true);
```