VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS INSTITUTAS


Rimgaudas Laucius

# KOMPILIATORIŲ INTERNACIONALIZACIJA

Daktaro disertacija
Technologijos mokslai, informatikos inžinerija (07 T)

Vilnius, 2007

Disertacija rengta 2002–2006 metais Matematikos ir informatikos institute.
Disertacija ginama eksternu.
Mokslinis konsultantas:
Doc. Dr. Valentina DAGIENĖ (Matematikos ir informatikos institutas, technologijos mokslai, informatikos inžinerija – 07T).

TURINYS

# LENTELIŲ SĄRAŠAS



# PAVEIKSLŲ SĄRAŠAS

# I. ĮVADAS

## 1.1. Mokslo problemos aktualumas

Informacinės technologijos tapo žmonių kasdienybės dalimi ir jau keistai atrodo dar neseniai gyvavusi nuomonė, kad norėdami jomis naudotis turime išmokti anglų kalbą. Nors vis dar daugiausia programinės įrangos (PĮ) sukuriama JAV ir jos pirminė kalba būna anglų, tačiau kuo toliau tuo didesnę PĮ rinkos dalį sudaro užsienio šalys. Tam, kad PĮ turėtų pasisekimą užsienio šalių rinkose, būtina ją lokalizuoti. Tačiau PĮ lokalizavimas beveik neįmanomas prieš tai jos neinternacionalizavus.

PĮ internacionalizavimas yra gamintojo prerogatyva ir tai yra PĮ gamybos proceso dalis. Todėl didelę įtaką jam turi gamybai naudojamų priemonių internacionalizacijos lygis. Jei priemonės nėra pakankamai internacionalizuotos, tuomet šis procesas yra neįmanomas arba reikalauja didesnių papildomų investicijų. Pavyzdžiui, akivaizdu, kad programuotojas susidurs su sunkumais kurdamas internacionalizuotą PĮ, jei programavimo priemonės neleidžia pirminiame tekste naudoti daugiakalbio teksto.

Ankstesni PĮ lokalizavimo darbai [DL03] [La03] [DL04] ir *Free Pascal* kompiliatoriaus pritaikymo Lietuvos mokykloms [DL01] [La01] srityse atskleidė, kad lokalizuojant PĮ vis dar išskyla daugybė problemų, kurių priežastimi yra nepakankamas jos internacionalizacijos lygis. Daugelis autorių linkę šių priežasčių ieškoti PĮ gamybos (internacionalizavimo) procese [Yo01] [Ye03] [Su01]. Tačiau pagrindinės priežastys kyla iš giliau t. y. iš nepakankamo PĮ gamybos priemonių internacionalizacijos lygio.

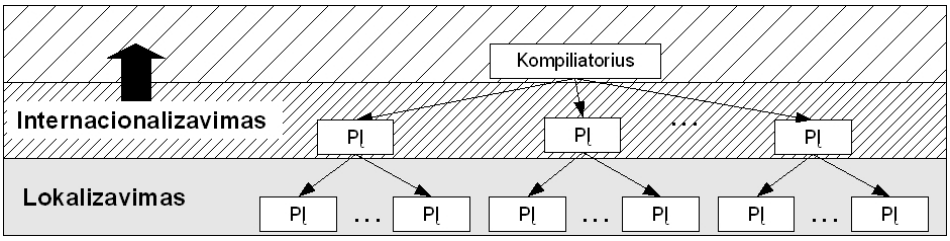
Dauguma PĮ yra sukuriama priemonėmis, kurių pagrindinė dalis yra kompiliatorius. Todėl tolesni tyrinėjimai buvo nukreipti būtent kompiliatorių internacionalizacijos kryptimi, ir šis darbas tapo logišku ankstesnių darbų tęsiniu.

Darbas aktualus ir praktiniu, ir moksliniu požiūriu dėl kompiliatorių internacionalizavimo ir kompiliatorių internacionalizuotumo lygio vertinimo metodų trūkumo. Darbe pateiktas kompiliatorių internacionalizavimo metodas padės spręsti egzistuojančias kompiliatorių ir jais kuriamos programinės įrangos internacionalizavimo problemas. Šio darbo rezultatai aktualūs visiems PĮ gamintojams ir pateiktos žinios gali būti panaudotos kuriant kompiliatorius.

Darbe pateiktas kompiliatorių internacionalizuotumo lygio įvertinimo metodas leis įvertinti kompiliatorių internacionalizuotumo lygį. Tai leis įvertinti kompiliatorių galimybes kurti internacionalizuotą PĮ, juos palyginti internacionalizuotumo aspektu.

## 1.2. Hipotezės

- Šiuo metu naudojami kompiliatoriai nėra pakankamai internacionalizuoti.
- Programinės įrangos internacionalizavimo sąnaudas galima sumažinti jos gamybai naudojant internacionalizuotą kompiliatorių.
- Dalį programų internacionalizavimo elementų galima perkelti į aukštesnį lygį – kompiliatorių internacionalizavimą.




*1 pav*. Internacionalizacijos elementų perkėlimas į kompiliatorių internacionalizacijos lygį

Pavykus perkelti internacionalizacijos elementus į kompiliatorių internacionalizacijos lygį (1 pav.) butų galima pakelti jais gaminamos PĮ internacionalizacijos lygį. Tai atneštu didelės naudos nes vieną kartą internacionalizavus kompiliatorių, jį galima daug kartų panaudoti PĮ kūrimui, kurios internacionalizavimui reiktų mažesnių sąnaudų, nei sukūrus ją neinternacionalizuotu kompiliatoriumi.

## 1.3. Darbo tikslas

Išanalizuoti kompiliatorių internacionalizacijos elementus, sukaupti ir susisteminti kompiliatorių internacionalizacijos žinias ir pateikti metodą, kuriuo remiantis būtų kuriama internacionalizuota kompiliatorių programinė įranga.

## 1.4. Darbo uždaviniai

1. Išalizuoti mokslinę ir metodinę literatūrą programinės įrangos ir kompiliatorių internacionalizavimo klausimais, aptarti teorinius internacionalizavimo aspektus.
2. Internacionalizacijos aspektu aptarti ir palyginti kelis daugiausia naudojamus kompiliatorius.
3. Ištirti kompiliatorių internacionalizavimo aspektus ir juos susisteminti.
4. Sukurti ir aprašyti kompiliatorių internacionalizavimo metodą.
5. Eksperimentiškai, dalinai internacionalizuoti pasirinktą kompiliatorių.

## 1.5. Tyrimų metodai

Darbe naudoti šie mokslinio darbo metodai: 1) informacijos paieška, sisteminimas, analizė ir apibendrinimas, 2) atvejų analizė, 3) testas, 4) lyginamoji duomenų analizė, 5) eksperimentas.

Informacijos paieškos, sisteminimo, analizės ir apibendrinimo metodai naudoti siekiant sukaupti ir pateikti kompiliatorių internacionalizavimo srities žinias.

Atvejų analizės ir testo metodai naudoti atliekant kompiliatorių internacionalizuotumo tyrimą.

Lyginamosios duomenų analizės metodas naudotas analizuojant kompiliatorių internacionalizuotumo tyrimo metu gautus rezultatus.

Eksperimento metodas naudotas internacionalizuojant *Free Pascal* kompiliatorių.

## 1.6. Mokslinis naujumas

Kompiliatorių internacionalizacijos tema iš esmės nauja – egzistuoja tik netiesiogiai su šia tema susiję mokslinės informacijos šaltiniai, nagrinėjantys bendrosios paskirties PĮ internacionalizaciją. PĮ internacionalizacija daugiau susidomėta praeitame dešimtmetyje, kai atsirado toks poreikis. Tačiau per visą šį laiką taip ir nebuvo atkreiptas deramas dėmesys į

kompiliatorių internacionalizaciją, nors tai ir yra vienas iš svarbiausių faktorių įtakojančių PĮ internacionalizavimą.

Dauguma šaltinių pateikia PĮ internacionalizavimą kaip procesą papildantį PĮ gamybos procesą ir bando spręsti internacionalizavimo problemas ieškant efektyvesnių, kokybiškesnių šio proceso modelių [Yo01] [Ye03] [Su01] ir metodų [HH97] [Dr03] [Ka95] ir daugelis kitų). Tačiau tai problemų iš esmės neišsprendžia, PĮ internacionalizacija išlieka sudėtingu, daug investicijų reikalaujančiu procesu. Iš esmės problemas derėtų spręsti nuo pradinės PĮ gamybos pakopos, t. y. įdiegiant PĮ internacionalizavimo metodus PĮ gamybai naudojamose priemonėse. Tai pakeistų ir patį požiūrį į PĮ internacionalizavimo procesą – jis taptų ne papildoma, bet neatskiriama, integruota, savaimine PĮ gamybos dalimi.

Šiame darbe pateikiamas naujai suformuotas metodas kompiliatorių internacionalizuotumo lygiui įvertinti. Juo remiantis išnagrinėti kompiliatorių internacionalizuotumo lygiai ir nustatyta įtaka jais kuriamos programinės įrangos internacionalizuotumui. Pateiktas kompiliatorių internacionalizavimo metodas paremtas susistemintomis, naujausiomis kompiliatorių internacionalizavimo srities žiniomis, sukauptomis teorinės analizės, atliktų kompiliatorių internacionalizuotumo lygio tyrimo ir eksperimentinio kompiliatoriaus internacionalizavimo metu.

## 1.7. Praktinė vertė

Disertacijos aktualumas remiasi praktiniais poreikiais. Pateikta metodika ir sukauptos žinios galės būti naudojamos kompiliatorių internationalizavimui ir padės spręsti PĮ internacionalizavimo problemas.

Disertacija taip pat aktuali moksliniu požiūriu. Jame pateikiamas naujas, moksliškai pagrįstas metodas skirtas kompiliatorių internacionalizuotumo lygio įvertinimui.

## 1.8. Aprobavimas

Disertacijos rezultatai pateikti 8 mokslinėse publikacijose:

- 2 publikacijos išspausdintos leidiniuose, įtrauktuose į Mokslinės informacijos instituto (ISI) duomenų bazę;
- 6 publikacijos išspausdintos leidiniuose įtrauktuose į periodinius ir tęstinius mokslo leidinius, registruotus kitose tarptautinėse mokslinės informacijos duomenų bazėse.

Disertacijos rezultatai pristatyti 12 konferencijų
Tarptautinėse konferencijose:

1. Internationalization of open source software: framework and some issues. *2nd International Conference Information Technology: Research and Education*. Londonas, Londono metropoliteno universitetas, 2004 m.
2. Free Pascal compiler internationalization. *LRC – X, The Global Initiative For Local Computing. Limerikas*, Lokalizacijos tyrimų centras, 2005 m.
3. *Free Pascal* application possibilities. *Teaching mathematics: retrospective and perspectives*. Vilnius, Vilniaus universitetas, 2005 m.
4. Issue of Selection of Programming Environment for Programming Curriculum in General Education. *Informatatics in Secondary Schools: Evolution and Perspectives*. Vilnius, Lietuvos Respublikos Seimas, 2006 m.

Vietinėse konferencijose:

5. *Free Pascal* panaudojimas informatikos kursui. *Lietuvos matematikų draugijos konferencija XLII*. Vilnius, Vilniaus Universitetas, 2001 m.
6. Raštinės programinės įrangos „OpenOffice.org“ adaptavimas lokalės normoms. *Kompiuterininkų dienos – 2003*. Vilnius, Lietuvos Respublikos Seimas, 2003 m.
7. Hiperteksto rašyklių palyginimas lokalizavimo galimybių požiūriu. *Lietuvos matematikų draugijos konferencija XLIV*. Vilnius, Vilniaus pedagoginis universitetas. 2003 m.
8. Lokalės, jų sandara ir ypatumai. *Informacinės technologijos 2003*. Kaunas, Kauno technologijos universitetas, 2003 m.
9. Programinė įrangos vertimo specifika ir dalinis automatizavimas. *Lietuvos matematikų draugijos konferencija XLV*. Kaunas, Lietuvos žemės ūkio universitetas, 2004 m.
10. *Free Pascal* kompiliatoriaus internacionalizavimas. *Kompiuterininkų dienos – 2005*. Klaipėda, Klaipėdos universitetas, 2005 m.
11. Lokalizavimo kurso projektavimas. *Lietuvos matematikų draugijos konferencija XLV*. Vilnius, Vilniaus universitetas. 2005 m.
12. *Free Pascal* programavimo sistema. *XIII pasaulio lietuvių mokslo ir kūrybos simpoziumas*. Vilnius, Mokslo tarybos rūmai, 2005 m.

## 1.9. Disertacijos struktūra

Disertacinį darbą sudaro įvadas, 4 dalys, literatūros sąrašas, autoriaus publikacijų sąrašas ir priedai. Diseracijos apimtis 88 puslapiai, 24 paveikslai, 10 lentelių ir 5 priedai (12 puslapių).

**Įvadas.** Tai įvadinis skyrius, kuriame pateikiama tyrimo tikslai ir uždaviniai, tyrimo aktualumas ir naujumas, tyrime naudoti metodai, gauti rezultatai, paskelbtos publikacijos.

**Teorinės tyrimo prielaidos.** Šiame skyriuje nagrinėjamos teorinės tyrimo prielaidos. Pirmojoje jo dalyje pateikiama internacionalizacijos sampratą. Tai viena iš pagrindinių darbe naudojamų sampratų. Internacionalizacijos dalykas yra gana naujas, todėl ir internacionalizacijos samprata yra gana nauja. Ji per savo trumpą egzistavimo laikotarpį daug keitėsi, todėl trumpai apžvelgiami pagrindiniai tyrimai bei straipsniai turėję įtakos jos raidai. Apžvelgiama kaip ją pateikia įvairūs autoriai.

Kitos dvi dalys yra analitinės. Jose apžvelgiami PĮ internacionalizuotumo vertinimo metodai. Nagrinėjamos jų ypatybės, apžvelgiami juos pateikiantys šaltiniai. Taip pat apžvelgiami šaltiniai, pateikiantys PĮ internacionalizuotumo vertinimo tyrimus, nagrinėjamos žemo internacionalizuotumo lygio priežastys ir pateikiamos išvados.

**Kompiliatorių internacionalizuotumo tyrimas.** Šiame skyriuje aprašoma tyrimų eiga ir pateikiami jų rezultatai.

Pirmojoje dalyje pateikiamas tyrimo objektas. Apžvelgiami kriterijai, lėmę konkrečių kompiliatorių kaip tyrimo objekto pasirinkimą. Pasirinkti kompiliatoriai tarpusavyje palyginami, apžvelgiant tyrimui reikšmingas jų savybes, bei sandarą.

Antrojoje dalyje pateikiamas tyrimo metodas. Šis metodas sukurtas remiantis panašiais PĮ internacionalizuotumo tyrimo metodais, nes kompiliatorių internacionalizuotumo tyrimų anksčiau dar nebuvo atlikta.

Trečiojoje dalyje apžvelgiami gauti tyrimo rezultatai. Išsamiau nagrinėjami pastebėti kompiliatorių internacionalizuotumo trūkumai bei jų priežastys.

Ketvirtojoje dalyje pateikiamos kompiliatorių internacionalizuotumo tyrimo išvados.

**Kompiliatorių internacionalizacijos metodas.** Šiame skyriuje pateikiamos kompiliatorių internacionalizavimo rekomendacijos.

Pirmoje dalyje apžvelgiami pagrindiniai kompiliatorių internacionalizavimo uždaviniai.

Antroje dalyje apžvelgiami kultūriniai ir kalbiniai faktoriai įtakojantys PĮ internacionalizaciją. Pagrindinis dėmesys skiriamas įvairių raštų palaikymo klausimams. Apžvelgiami pagrindiniai raštų skirtumai turintys įtakos internacionalizacijos realizacijai. Trumpai apžvelgiami kalbiniai PĮ internacionalizacijos elementai.

Trečioje dalyje apžvelgiami lokalių (tai vienas iš svarbiausių elementų PĮ internacionalizacijoje) standartai ir jų sandara. Lokalių standartai tarpusavyje palyginami. Trumpai apžvelgiami lokalių ir jų kultūrinių nuostatų registravimo standartai.

Ketvirtoje dalyje apžvelgiami duomenų kodavimo metodai. Pagrindinis dėmesys skiriamas rekomenduojamiems Unikodo standarto duomenų kodavimo metodams.

Penktoje dalyje apžvelgiami išteklių atskyrimo metodai. Jie tarpusavyje palyginami, pateikiami jų trūkumai ir pranašumai.

Šeštoje dalyje nagrinėjamas lokalizuotinų leksikos elementų internacionalizavimas. Pateikiamas rekomendacijos leksikos elementų atskyrimui, leidžiančios išspręsti kylančias lokalizuotų kompiliatorių tarpusavio suderinamumo problemas.

Septintoje dalyje pateikiamas internacionalizacijos karkasas. Jis sudaro rekomenduojamos principinės kompiliatorių internacionalizacijos pagrindą.

Aštuntoje dalyje nagrinėjama internacionalizacijos karkasą sudarančių komponentų realizacija.

**Eksperimentinis kompiliatoriaus internacionalizavimas.** Šiame skyriuje pristatomas eksperimentas atliktas internacionalizuojant kompiliatorių.

Pirmoje dalyje apžvelgiami eksperimento tikslai.

Antroje dalyje apžvelgiamas eksperimento objektas.

Trečioje dalyje apžvelgiama *FPS* sistema, sukurta remiantis šio darbo rezultatais, kurioje buvo įdiegtas internacionalizuotas *Free Pascal* kompiliatorius.

Ketvirtoje dalyje apžvelgiami išplėtimai atlikti internacionalizuojant *Free Pascal* kompiliatorių, pateikiama eksperimento metu sukaupta patirtis ir gauti rezultatai

Penktoje dalyje pateikiamos kompiliatoriaus internacionalizavimo eksperimento išvados.

# II. TEORINĖS TYRIMO PRIELAIDOS

Šiame skyriuje apžvelgiami informacijos šaltiniai, susiję su kompiliatorių internacionalizacija ir jos tyrimais. Pagrindiniai apžvalgos tikslai: 1) tiksliau apibrėžti kas tai yra kompiliatorių internacionalizacija; 2) apžvelgti ką šioje srityje jau yra nuveikę kiti tyrinėtojai ir pabandyti tiksliau nustatyti darbo probleminę sritį.

Kompiliatorių internacionalizacijos tema iki šiol yra nauja ir šaltinių, kuriuose ji būtų nagrinėjama iš esmės, nėra. Tačiau ji iš dalies susijusi su bendros paskirties PĮ internacionalizacija, todėl darbe daug kur remiamasi šią temą nagrinėjančiais šaltiniais.

PĮ internacionalizavimas pakankamai naujas dalykas, didelę svarbą įgavęs tik praeitame dešimtmetyje. Pastaruoju metu jis buvo gana intensyviai nagrinėjamas – buvo publikuota nemažai informacijos šaltinių. Juos grubiai galima suskirstyti į dvi grupes – nagrinėjančius: 1) tarpkultūrinius skirtumus įtakojančius PĮ internacionalizaciją; 2) PĮ internacionalizavimo procesą. Abiejų tipų šaltiniai glaudžiai tarpusavyje susiję, nes nagrinėjant PĮ internacionalizaciją, svarbu žinoti nuo kokių kultūrinių skirtumų ji yra priklausoma.

Ieškant informacijos šaltinių daugiausia naudotasi tarptautinėmis mokslinių informacijos šaltinių duomenų bazėmis. PĮ internacionalizacijos tema gana plati, todėl dauguma mokslinių šaltinių nagrinėja konkrečią, siaurą jos sritį. Tokiu būdu jie leidžia giliau pažvelgti į atskirus nagrinėjamos temos klausimus. Darbe taip pat remiamasi PĮ internacionalizavimą nagrinėjančia profesine literatūra, kurioje pateikiamas platesnis ir praktiškesnis požiūris nei mokslinėse publikacijose.

Ne visi surasti informacijos šaltiniai pasirodė vienodai vertingi. Pagrindiniai pastebėti jų trūkumai:

1. Visapusiškumo stoka. Pavyzdžiui, nagrinėjama PĮ internacionalizacija konkrečios operacinės sistemos (OS) aplinkoje. Tokiu būdu apsiribojama tik ta internacionalizacijos samprata, kurią ta sistema realizuoja ir tyrimų rezultatai nėra universalūs.
2. Naujumo stoka. PĮ, jos gamybos technologijos bei priemonės labai sparčiai vystosi, todėl didelė dalis šaltinių, net ir paskelbtų neseniai, jau pateikia pasenusią informaciją. Tai taip pat susiję su prisirišimu prie konkrečių sistemų.

Be minėtų šaltinių darbe taip pat remiamasi standartais ir jiems ekvivalenčiais dokumentais. Šiame darbe jie žymimi jų oficialiomis pavadinimų santrumpomis arba numeriais, o jų pilnų pavadinimų sąrašas pateiktas 1 priede.

## 2.1. Internacionalizacijos samprata

Įvairių autorių pateikiami PĮ internacionalizacijos apibrėžimai šiek tiek skiriasi. Publikacijose dažnai remiamasi LISA[1] pateiktu apibrėžimu: *internacionalizacija yra produkto apibendrinimo procesas tam, kad jis galėtų apdoroti (palaikyti) skirtingas kalbas ir kultūrines nuostatas jo neperprojektavus*[2] [LISA06]. Panašiai internacionalizacija apibrėžta ir enciklopediniame kompiuterijos žodyne: *programinės įrangos ir jos duomenų struktūrų projektavimas taip, kad visa tai būtų galima lengvai adaptuoti įvairioms kalboms ir kultūroms* [DGJ05].

[1] Lokalizacijos industrijos standartų associacija. Angl. *The Localisation Industry Standards Association.*
[2] Angl. *Internationalization is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for re-design.*

Šie apibrėžimai pateikia internacionalizaciją kaip tam tikrą procesą. Tačiau remiantis lietuvių kalbos taisyklėmis procesui žymėti derėtų naudoti **internacionalizavimo** terminą, o internacionalizacijos terminą apibrėžti iš naujo. Šiame darbe **internacionalizaciją** apibrėšime kaip: 1) *programinės įrangos savybių darančių ją lengvai adaptuojamą įvairioms kalboms ir kultūroms visuma*; 2) *dalykas apimantis ir nagrinėjantis metodus bei technologijas, naudojamus sudarant programinės įrangos adaptavimo įvairioms kalboms ir kultūroms galimybes.*

Internacionalizacija yra artimai susijusi su lokalizacija. Nes **lokalizavimas** yra *programinės įrangos pritaikymas tam tikrai kalbinei ir kultūrinei aplinkai* [DGJ05]. Jei neplanuojama PĮ lokalizuoti tai nebūtina ją ir internacionalizuoti, ir atvirkščiai PĮ neįmanoma lokalizuoti jei ji neinternacionalizuota.

Internacionalizacijos ir lokalizacijos raida vyko kartu. Jos pradžia laikomas praeitas dešimtmetis. Ši raida daugiausia susijusi su PĮ gamintojų noru gauti daugiau pelno išplečiant savo produktų rinkas į kitas šalis. Pradžioje PĮ daugiausia naudojosi tik kompiuterių specialistai ir PĮ rinka buvo nedidelė. Todėl siekiant įtikti kitų šalių pirkėjams užtekdavo išversti PĮ dokumentaciją ir tai buvo laikoma jos lokalizavimu. Pati PĮ buvo verčiama tik pavieniais atvejais. Vėliau, paplitus asmeniniams kompiuteriams, PĮ lokalizavimo poreikis išaugo. Vartotojai teikė pirmenybę lokalizuotai PĮ, nes ji buvo praktiškesnė, darbas su ja efektyvesnis. Lokalizavimas buvo pradėtas nuo pačių būtiniausių dalykų – PĮ pritaikymo apdoroti duomenis, atitinkančius vietinius standartus, kultūrines nuostatas, raštą. Dialogo kalba verčiama į kitas kalbas buvo ne visada [Es03].

Iš pradžių lokalizavimas buvo atliekamas pirminio programos teksto lygmenyje. Tačiau netrukus toks metodas pasirodė esąs labai neefektyvus ir nepatikimas. Reikalingas išversti teksto eilutes, išbarstytas po visą pirminį programų tekstą, būdavo sunku surasti. Taisyti, kompiliuoti ir derinti vienu metu tekdavo kelis skirtingų lokalių PĮ pirminio teksto variantus. Šie trūkumai vertė ieškoti programavimo metodų, kad lokalizavimas būtų paprastesnis ir pigesnis. Imta atskirti lokalizuojamus išteklius nuo veiksmus aprašančio programos pirminio teksto ir toks programavimo metodas imta vadinti internacionalizavimu.

Internacionalizacijos atsiradimo pradžioje pagrindiniai jai kelti uždaviniai buvo: sudaryti galimybę PĮ tekstus versti į kitas kalbas, teisingai įvesti ir išvesti duomenims, teisingai operuoti su kultūrines nuostatas atitinkančiais duomenų formatais, pavyzdžiui, piniginiais vienetais, datomis, skaičiais ir t. t. Vystantis PĮ, internacionalizacijos samprata kito ir jai keliamų uždavinių daugėjo. Vienas iš pirmųjų dažnai cituojamų šaltinių, turėjusių didelę įtaką PĮ internacionalizacijos sampratos raidai buvo Russo ir Boor straipsnis [RB93]. Jame teigiama, kad siekiant PĮ paklausos tarptautinėje rinkoje, jos gamintojai turi susipažinti su tarpkultūriniais skirtumais ir į juos atsižvelgti, tuo pačiu pakeisdami iki tol gyvavusį PĮ gamybos procesą – į jos gamybą įtraukdami internacionalizavimą. Pažymima, jog visiškai internacionalizuotos PĮ gamybos procesas neturi apsiriboti vien galimybių teksto vertimui, skirtingų datų, laiko, skaičių formatų ir panašių elementų palaikymui sudarymu. Pateikiamas nuo kultūros priklausomų PĮ elementų (susijusių su tekstų vaizdavimu, formatais, paveikslais, spalvomis, PĮ veiksena ir funkcionalumu, ir pan.) sąrašas su komentarais. Taip pat svarbu pažymėti, kad PĮ gamybos ciklas papildomas internacionalizacijos testavimu prieš išleidžiant pradinę jos versiją. Anksčiau tai nebuvo daroma.

Russo ir Boor [RB93] buvo vieni pirmųjų, kurie atkreipė dėmesį į tai kad PĮ praktiškumui įtakos turi gana plati aibė nuo kultūros priklausomų PĮ faktorių. Vėliau Yeo [Ye96] juos suskirstė į dvi grupes: akivaizdžius[3] ir neakivaizdžius[4] kultūrinius faktorius. Akivaizdūs

[3] Angl. *Overt*
[4] Angl. *Covert*

kultūrinius faktoriai (pvz. matavimo vienetai, kalendoriai) jau yra pakankamai gerai apibrėžti ir lengvai suprantami. Tuo tarpu neakivaizdūs faktoriai (pvz. grafinė, akustinė ir vizualinė informacija, spalvos, funkcionalumas, metaforos) yra nepakankami apibrėžti arba išvis neapibrėžti, nors yra gana svarbūs gaminant internacionalizuotą PĮ. Straipsnyje pateikiama jų klasifikacija, kuri galėtų būti pagrindu sudarant, PĮ gamintojams skirtas, kultūrinės informacijos duomenų bazes.

Kituose šaltiniuose [Gr97] analogiškai PĮ internacionalizacija skirstoma į du lygmenis paviršinį ir kultūrinį. Paviršinis lygmuo susijęs su akivaizdžiais kultūriniais faktoriais ir yra pakankamai aiškus ir santykinai nesunkiai realizuojamas, o giluminis lygmuo, susijęs su spalvų, paveikslų ir kitų mažiau formalių elementų pritaikymu vis dar kelia internacionalizavimo ir lokalizavimo problemų, nes tie elementai nėra pakankamai gerai ištirti ir apibrėžti. [Sh00] šiuos lygmenis vaizduoja kaip aisbergą, kurio paviršinis lygis yra matomas ir pakankamai aiškus, tačiau, kaip ir aisbergo, pagrindinę ir kur kas mažiau išnagrinėtą internacionalizacijos dalį sudaro kultūrinis lygis.

Tiriant kultūros įtaką PĮ praktiškumui buvo nustatyta daugiau PĮ internacionalizacijai svarbių kultūrinių faktorių. Evers [Ev01] disertacijoje pateikiamas tyrimas, kurio tikslas įvertinti naudotojo kultūrinės patirties įtaką PĮ sąsajos suvokimui. Nustatyta, kad metaforiškai išreikštos PĮ sąsajos elementus skirtingų kultūrų respondentai gali asocijuoti su savo aplinkos objektais, kurie gali neatlikti gamintojo užkoduotų objektų ir dėl to gali nukentėti PĮ praktiškumas. Tyrimas taip pat parodė, kad tam pačiam tikslui pasiekti skirtingose kultūrose gali būti priimta naudoti skirtingus veiksmus. Todėl PĮ internacionalizavimas neturi apsiriboti vien tik galimybių PĮ sąsajos lokalizavimui sudarymu, bet gali apimti ir PĮ atliekamus veiksmus (apie tai pamiršta daugelis autorių rašančių apie PĮ internacionalizaciją). Kersten su bendraautoriais [KKR01] akcentuoja keletą PĮ tipų, kuriai tai gali būt ypač aktualu bei pateikia idėjų kaip tokiais atvejais gali būti realizuojama PĮ internacionalizaciją.

Chong [Ch98] tyrimas pademonstravo, jog internacionalizuojant PĮ tikslinga atsižvelgti į skirtingose kultūroje nusistovėjusius suvokimo bei mastymo stilius. Šio tyrimo metu nustatyta, kad Kinijos kompiuterių naudotojų (jiems labiau būdingi konkretaus suvokimo ir tematinio mastymo stiliai) ir JAV naudotojų (būdingi abstraktaus suvokimo ir funkcinio mastymo stiliai) PĮ naudojimo efektyvumas skiriasi, kai informacija pateikiama abstrakčia arba konkrečia išraiška, o sąsaja realizuojama funkciniu arba tematiniu būdu. Rezultatai parodė, kad naudotojai efektyviau naudojasi ta PĮ kuri atitinka jų suvokimo bei mastymo stilius.

Net keletas tyrimų [FG03] [Ku98] [AG00] patvirtina arba iš dalies patvirtina Hofstede [Ho91] apibrėžtų kultūrinių dimensijų (galios santykio, individualizmo prieš kolektyvizmą, vyriškumo prieš moteriškumą, neapibrėžtumų vengimo, ilgalaikės perspektyvos prieš trumpalaikę) reikšmingą įtaką PĮ praktiškumui. Pavyzdžiui, Aaron ir Gould [AG00] analizuoja skirtingoms kultūroms atstovaujančių šalių tinklalapius ir pateikia tai patvirtinančių pavyzdžių.

Keičiantis supratimui apie kultūros įtaką PĮ praktiškumui, kito ir internacionalizacijos samprata. Iš pradžių buvo laikoma, kad PĮ internacionalizuotumas reiškia jos neutralumą kultūrinių nuostatų atžvilgiu. T. y. lokalizuojant tokią PĮ turi pakakti tik jos vertimo. Tačiau, kaip pažymi Kersten su bendraautoriais [KKR00] ir kiti specialistai, neegzistuoja universalios kultūros arba jos pagrindų, todėl negalima sukurti PĮ, kuri savaime būtų pritaikyta visų kultūrų atstovams. Anksčiau minėti tyrimai patvirtino, kad visiškas PĮ adaptavimas konkrečiai kultūrai turi reikšmingą teigiamą įtaką jos praktiškumui. Tuo tarpu šis internacionalizavimo būdas praktiškumą aukoja vardan mažesnių investicijų, todėl jis laikytinas ydingu [Vi02].

Viena iš internacionalizacijos sampratų remiasi idėja, kad internacionalizuota PĮ turi pati automatiškai prisitaikyti konkrečios kultūrinės terpės nuostatų atžvilgiu. Tačiau sukurti PĮ kuri

pilnai prisitaikytų neįmanoma, nes neįmanoma formaliai apibrėžti visų nuo kultūros priklausomų PĮ elementų. Pavyzdžiui, neįmanoma formaliai apibrėžti paveikslų taip, kad juos būtų galima automatiškai pritaikyti kitai kultūrinei aplinkai. Todėl šią idėją reikėtų vertinti kaip siekiamybę ir ją galima laikyti ateities iššūkiu, nes šiuo metu dar neegzistuoja duomenų bazių, kuriose būtų sukaupti išsamūs įvairių kultūrų nuostatų aprašai bei PĮ internacionalizavimo metodų, kurie leistų pasiekti aukštą tokios internacionalizacijos lygį.

Lokalizacijos kokybei pastaruoju metu keliamus reikalavimus neformaliai galima apibūdinti Schäler [Sh02] citata: *PĮ turi būti adaptuota tam tikrai kalbinei ir kultūrinei terpei taip, lyg ji būtų sukurta toje terpėje*[5]. Tai pasiekti galima tik adaptavus PĮ atsižvelgiant į visas konkrečioje terpėje galiojančias kalbines ir kultūrines nuostatas, todėl PĮ internacionalizavimo uždaviniai – sudaryti visas galimybes, kad toks adaptavimas būtų įmanomas.

## 2.2. PĮ internacionalizuotumo vertinimo metodų apžvalga

PĮ internacionalizavimo ir kompiliatorių internacionalizavimo uždaviniai skiriasi (žr. 4.1 skyrelį), todėl skiriasi ir jų internacionalizuotumui keliami reikalavimai. Todėl PĮ internacinalizuotumo lygio vertinimo metodai netinka kompiliatorių internacionalizuotumui vertinti. Tačiau kuriant metodą kompiliatorių internacionalizuotumui vertinti gali būti atsižvelgta į jau egzistuojančią PĮ internacionalizuotumo vertinimo pratiką, todėl ją trumpai apžvelgime.

PĮ gali būti internacionalizuota įvairiais lygiais. Pavyzdžiui, PĮ skirtos vietinei rinkai, t. y. kurios nenumatoma lokalizuoti, internacionalizuoti neapsimoka, nes tai reikalauja papildomų investicijų, kurios neatneš naudos. Tačiau jei PĮ skirta tarptautinei rinkai ją internacionalizuoti būtina. Deja, gamintojai dažnai, dėl įvairių priežasčių neinternacionalizuoja arba nepakankamai internacionalizuoja PĮ ir šiuo atveju. Dažniausiai tai lemia noras išsiversti be internacionalizavimo, nes jis reikalauja nemažų investicijų. Be to vis dar egzistuoja klaidinga nuomonė, kad PĮ lokalizavimas pačių lokalizuotojų reikalas. Tuo, kad didelė dalis tarptautinei rinkai skirtos PĮ yra internacionalizuota nepakankamai galima įsitikinti atlikus PĮ internacionalizuotumo tyrimus.

Young [Yo01], Hall ir Hudson [HH02] ir kituose darbuose akcentuojama, kad internacionalizuotumo vertinimas turėtų būti viena iš PĮ gamybos proceso dalių. Įvertinus internacionalizuotumą, rastus trūkumus galima būtų pašalinti to paties gamybos proceso metu ir jie sukeltų mažiau rūpesčių, nei pastebėti jau pradėjus lokalizuoti PĮ, nes tuomet jų šalinimas yra gerokai sunkesnis.

Universalios metodikos kaip įvertinti PĮ internacionalizacijos lygį neegzistuoja. Kaip ir priemonių, kurios leistų internacionalizacijos lygį įvertinti automatiškai. Egzistuoja tik pagalbinės priemonės, kurios gali būti naudojamos šiam darbui palengvinti. Pavyzdžiui, naudojant priemones, kurios leidžia atlikti pseudo-vertimą (verčiamas teksto eilutes pakeičia automatiškai sugeneruotomis teksto eilutėmis sudarytomis iš nurodytai lokalei būdingų rašto ženklų), galima ištirti ar PĮ pakankamai gerai parengta vertimui. Taip pat egzistuoja priemonės skirtos daliai kitų su internacionalizacija susijusių elementų patikrinimui. Pavyzdžiui, valdiklių langų išdėstymo (pavyzdžiui, lygiavimo arba persidengimo), sparčiųjų klavišų komandų persidengimo, teksto išėjimo už valdiklių langų ribų tikrinimui [Es02].

Paprastai internacionalizuotumo lygis nustatomas tikrinant PĮ atitikimą reikalavimams pagal specialiai tam sudarytus klausimynus. Šie klausimynai sudaromi taip, kad išsamiai apimtų nagrinėjamos PĮ internacionalizacijos aspektus. Jais remiantis atliekamas ne tik kiekybinis, bet ir

[5] Angl. Citata. *Adapt products so that they look and feel as if they were developed in country, the language and culture that they are localized for.*

kokybinis vertinimas. T. y. paprastai klausimai sudaromi taip, kad į juos būtų galima atsakyti vienu iš galimų variantų: „taip“, „ne“, „nepakankamai“, „netaikoma“, o klausimų įvertinimas „nepakankamai“ yra pateikiamas su pastabomis paaiškinančiomis pastebėtus trūkumus.

Didesni PĮ gamintojai PĮ internacionalizacijos vertinimams atlikti yra susikūrę ir naudoja savus metodus. Kai kurie gamintojai viešai pateikia klausimynus kuriais remiantis galima įvetinti PĮ sukurtos jų priemonėmis arba skirtos jų platformai internacionalizacijos lygį. Pavyzdžiui, *Sun* pateikia *Java*[6] pagrindu sukurtos PĮ internacionalizacijos vertinimo metodus [Su01] [Su06], Kano [Ka95] ir Microsoft [Mi06] pateikia klausimynus skirtus *Windows*[7] OS šeimai skirtos PĮ, *Apple*[8] [Ap06] – *Mac OS* skirtos PĮ internacionalizacijos vertinimui.

PĮ internacionalizuotumo lygis yra svarbus rodiklis lokalizavimu užsiimančioms kompanijoms. Internacionalizuotumo įvertinimas leidžia dar prieš lokalizuojant PĮ numatyti galimus sunkumus, apskaičiuoti lokalizavimo kainą ir pan. Kai kurios kompanijos pateikia jų naudojamus PĮ internacionalizacijos vertinimo metodus viešai. Pavyzdžiui, Rätzmann ir Young [RY03] aprašo LISA rekomenduojamą internacionalizacijos vertinimo metodą, Rubic [Ru06] pateikia jų naudojamus klausimynus.

Tiek PĮ gamintojų, tiek lokalizuotojų pateikiami metodai yra nepakankamai išsamūs ir netinka moksliniam tyrimui. Gamintojų naudojamuose metoduose dažniausiai atsižvelgiama tik į jiems aktualius internacionalizacijos aspektus. Pavyzdžiui, į tuos aspektus, kurių palaikymą realizuoja jų PĮ arba jos naudojama platforma. Lokalizavimo kompanijos siekdamos lokalizavimą atlikti greitai ir pigiai dažniausiai neatsižvelgia į mažiau reikšmingus arba sunkiau realizuojamus internacionalizacijos aspektus. Tuo tarpu moksliniam tyrimui skirtas metodas turi būti visapusiškas, jame turi būti atsižvelgta ne tik į dabartinę PĮ gamybos ir lokalizavimo praktiką, bet ir į perspektyvas bei tendencijas.

## 2.3. PĮ internacionalizuotumo lygio apžvalga

Vykstant globalizacijai PĮ rinkos smarkiai išsiplėtė. Netgi nedidelių programų rinkos interneto dėka gali apimti kone visą pasaulį. Remiantis įvairiais šaltiniais [Es03] [Ye01] daugiau nei pusę pelno didieji JAV PĮ gamintojai gauna ir užsienyje parduodamos produkcijos. Tačiau PĮ internacionalizacijos lygio tyrimai rodo, kad jos lygis dažnai yra gana žemas. Todėl apžvelgsime šiuos tyrimus ir pabandysime nustatyti šio reiškinio priežastis.

Pradėsime nuo atliktų tyrimų pavyzdžių. Käpyaho [Ka01] atliktas delniniams kompiuteriams skirtų OS internacionalizacijos lygio tyrimas atskleidė, kad visos tirtos OS turi didesnių ir mažesnių trūkumų. Tyrimo metu geriausiai įvertintos Microsoft *Windows* CE 3.0 ir Symbian 6.0 OS[9] surinko atitinkamai 31 ir 30 balų iš 39 galimų. Na, o kitos OS buvo įvertintos dar prasčiau, pavyzdžiui Palm OS 3.5[10] surinko 10, Java 2 Platform, Micro Edition (+Foundation profile)– 24 balus.

Hiperteksto rašyklių internacionalizuotumo tyrimas [La03] atskleidė, kad tik 5 iš 21 tirtų rašyklių tenkina pagrindinius internacionalizuotumo reikalavimus. Na, o detalesnis jų tyrimas parodė, kad nei viena iš jų nėra pakankamai gerai internacionalizuota.

[6] „Java“ yra Sun Microsystems bendrovės registruotas prekinis ženklas
[7] „Windows“ yra Microsoft bendrovės registruotas prekinis ženklas
[8] „Apple“ ir „Mac“ yra Apple Computer bendrovės registruoti prekiniai ženklai
[9] „Symbian“ yra Symbian Ltd. bendrovės registruotas prekinis ženklas
[10] „Palm“ yra PalmSource bendrovės registruotas prekinis ženklas

Pasaulyje plačiai naudojamo atvirųjų biuro programų paketo OpenOffice.org tyrimas [LD04] parodė, kad jos internacionalizavimui pasirinktas neefektyvus, pasenęs metodas, dėl ko ženkliai padidėja jos lokalizavimui reikalingos išlaidos.

Immonen ir Sajaniemi [IS03] PĮ internacionalizavimo Soumijoje tyrimo ataskaitoje teigiama, kad dauguma apklausoje dalyvavusių Suomijos PĮ gamintojų supranta internacionalizacijos svarbą jų gaminamai PĮ, tačiau daugelis apsiriboja tik galimybių jos tekstų vertimui sudarymu. Kitų PĮ elementų adaptavimui galimybes sudaro tik keli gamintojai ir tai tik retais atvejais. Interviu metu įvairūs gamintojai nurodė gana įvairias priežastis lemiančias žemą jų gaminamos PĮ internacionalizacijos lygį. Jas galima suskirstyti į dvi grupes 1) pakankamai kvalifikuotų PĮ internacionalizavimo specialistų trūkumas 2) techninės priežastys. Reikia pažymėti, kad gamintojai labiau buvo akcentavo technines priežastis. Pavyzdžiui, buvo akcentuojamas internacionalizuotos PĮ rengimo standartų arba visuotinai priimtų metodų trūkumas, ženklų kodavimo problemos, nepakankamas internacionalizacijos palaikymas iš programavimo sistemų ir kt.

Problemos susiję su PĮ lokalizavimo arba internacionalizavimo specialistų rengimu apžvelgiamos mūsų straipsnyje [LD05]. Nors jame daugiausia akcentuojama Lietuvos situacija, tačiau remiamasi tarptautiniais tyrimais, kurie patvirtina, kad šių specialistų rengimo problemos egzistuoja ir daugelyje kitų šalių. Straipsnyje pažymima, kad PĮ projektuotojai ir programuotojai kvalifikaciją PĮ internacionalizavimo srityje turėtų įgauti aukštojo mokslo įstaigose papildomų kursų metu. Tačiau visame pasaulyje dar egzistuoja labai mažai aukštojo mokslo įstaigų į kurių programas jie būtų įtraukti. Todėl specialistų rengimo problema nuo seno sprendžiama pačių PĮ gamintojų papildomai apmokant darbuotojus bendrovės viduje [Es02]. PĮ internacionalizavimo kursų svarbą taip pat pažymi Hogan su bendraautoriais [HHP03], Sahama su bendraautoriais [SHH04] ir kt. specialistai.

Žemą PĮ internacionalizuotumo lygį dažnai lemia ir programuotojų nusistatymas. Trillo [Tr99] remdamasis Tognazzini [To92], Myers ir McCaulley [MM85], Sitton ir Chmelir [SC84] darbais pažymi, kad programuotojai teikia kitokius prioritetus projektuojamai PĮ nei paprasti naudotojai. Programuotojai dažnai tarsi pamiršta, kad kuria PĮ ne tik sau, bet ir kitiems naudotojams ir neatsižvelgia į jų poreikius. Gould ir Lewis [GL85] apklausus 450 IBM programuotojų nustatė, kad tik šeštas iš jų PĮ projektavimo stadijoje atkreipdavo dėmesį į naudotojų poreikius. Tuo tarpu tam, kad naudotojai galėtų maksimaliai efektyviai naudotis PĮ ir nepatirtų nepatogumo, būtina ją kurti atsižvelgiant ne tik į jų poreikius bet ir į kultūrinius skirtumus.

PĮ gamintojų nusistatymui prieš internacionalizavimą daugiausia įtakos turi su ja susijusios papildomos investicijos. Remiantis Hall ir Hudson [HH02] jos sudaro apie 10–20 % PĮ gamybos išlaidų. Šios išlaidos papildomai skiriamos įvairioms sritims: mokymui, gamybai, testavimui ir t. t. Tačiau pagrindinės priežastys dėl ko išlaidų procentas yra gana aukštas galiausiai susiveda į techninio pobūdžio problemas – menką internacionalizacijos palaikymą iš programavimo priemonių pusės. Nesant standartinių internacionalizavimo metodų, kuriuos realizuotų programavimo priemonės, gamintojai priversti patys ieškodami būdų jos internacionalizavimui. Tai taip pat lemia tai, kad daugiau laiko sugaištama projektuojam, gaminant ir testuojant PĮ, o internacionalizuotai PĮ sukurti būtini aukštą kvalifikaciją turintys šios srities specialistai.

Egzistuoja tiesioginis ryšys tarp programavimo priemonių internacionalizacijos lygio ir galimybių jomis kurti internacionalizuotą PĮ. Pavyzdžiui, akivaizdu, kad programuotojas susidurs su sunkumais kurdamas internacionalizuotą PĮ, jei programavimo priemonės neleidžia programoje naudoti daugiakalbio teksto.

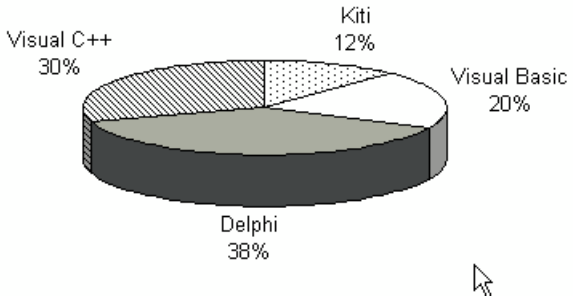
## 2.4. Išvados

1. Internacionalizavimas yra PĮ gamybos proceso dalis, todėl gamybos priemonių galimybės (internacionalizacija) jam turi didelę įtaką.
2. Įvairūs tyrimai atlikti PĮ internacionalizacijos srityje (tame tarpe ir šiame darbe pateiktas tyrimas) atskleidžia, kad dar egzistuoja gana daug problemų. Pagrindinės problemos yra techninio pobūdžio: internacionalizuotos PĮ rengimo standartų ir metodų trūkumas, ženklų kodavimo problemos, nepakankamas internacionalizacijos palaikymas iš programavimo priemonių ir kt. Dėl šių problemų PĮ internacionalizavimas reikalauja gana didelių investicijų – jos siekia 10–20 % visos PĮ gamybos išlaidų. Jos išauga dėl to, kad PĮ gamintojai yra priversti ieškoti individualių sprendimų susijusių su PĮ internacionalizavimu, o jų realizacija reikalauja gana didelio papildomo gamintojų indėlio.
3. Išspręsti daugumą šių problemų galima tik įdiegus sprendimus pačiame žemiausiame PĮ gamybos lygyje – PĮ gamybos priemonėse, t.y. jas internacionalizavus. Tokiu būdu internacionalizuotos PĮ gamyba taptų tokiu pat natūraliu procesu kaip šiuo metu gaminamos neinternacionalizuotos PĮ ir reikalautų žymiai mažesnių investicijų.

# III. KOMPILIATORIŲ INTERNACIONALIZUOTUMO LYGIO TYRIMAS

## 3.1. Tyrimo objektas

Siekiant, kad tyrime gauta informacija būtų kuo aktualesnė, tyrimo objektu buvo pasirinkti pasaulyje populiariausi kompiliatoriai. Jų populiarumui nustatyti buvo atliktas nedidelis tyrimas, kurio metu nustatyta kokiais kompiliatoriais sukurtos 50 atsitiktinai pasirinktų, *Windows* OS skirtų programų. Buvo parsiųsta po lygiai (po 25) nemokamų ir laikinai nemokamų programų. Ištyrus šias programas nustatyta, kad jų sukūrimui daugiausia buvo naudoti trys kompiliatoriai (įvairių versijų): *Delphi*[11], *Visual C++* ir *Visual Basic*[12]. Kompiliatorių pasiskirstymas pateiktas 2 paveiksle. Tyrimo rezultatai pateikti 2 priede.




*2 pav*. Kompiliatorių populiarumas kuriant *Windows* platformai skirtas programas

Taip pat buvo remtasi programavimo kalbų populiarumo tyrimais [We05] [Ti06]. Jie parodė, kad be tyrime nustatytų kompiliatorių, taip pat labai populiarus Java kompiliatorius. Mūsų atliktas kompiliatorių populiarumo tyrimas jo neatskleidė todėl, kad šis kompiliatorius daugiausia naudojamas žiniatinklio programoms veikiančiomis Java platformoje kurti.

*Visual C++* ir *Visual Basic* yra to paties PĮ paketo *Microsoft Visual Studio* dalys. Todėl tirti juos abu yra netikslinga ir buvo pasirinktas tik vienas iš jų – *Visual C++*.

Galiausiai buvo pasirinkti trys kompiliatoriai: 1) *Visual C++* (*Microsoft Visual Studio 2003 .NET*)[13], 2) *Java* (*Sun Java 1.5.0*)[14] ir 3) *Delphi* (*Borland Delphi 2005*).

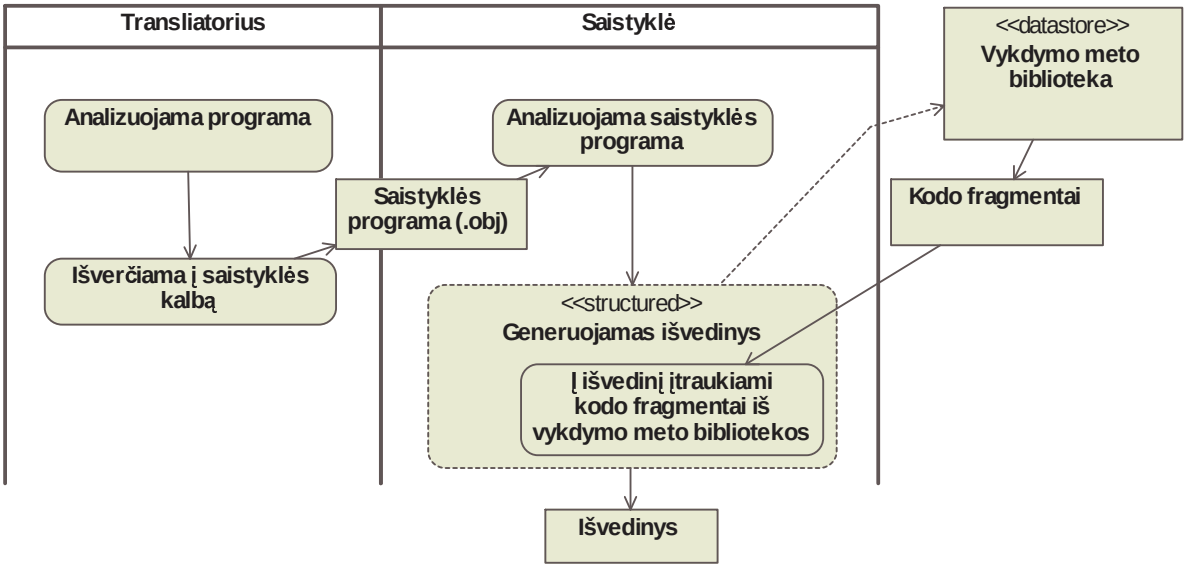
Visi kompiliatoriai skirti *Windows* OS, tačiau to nereikėtų laikyti apribojimu, nes tiriamos, kompiliatorių internacionalizuotumo savybės nesusiję su OS.

Nagrinėjami kompiliatoriai standartizuoti skirtingais lygiais. Labiausiai standartizuotas *Visual C++* kompiliatorius. Remiamasi tarptautiniais C (ISO/IEC 9899) ir C++ (ISO/IEC 14882) programavimo kalbų standartais (nors jis, daugiausia dėl įvairių išplėtimų nėra pilnai su jais suderinamas). Pažymėtina, kad C++ standartas yra nuolat atnaujinamas, todėl atspindi šiuolaikinius C++ kompiliatorių gamintojų ir programuotojų poreikius. Standartas sudarytas iš dviejų dalių. Pirmoji dalis skirta programavimo kalbą, antroji – standartinei bibliotekai. Standartinė biblioteka iš esmės atitinka *Visual C++* kompiliatoriaus vykdymo meto biblioteką (VMB).

[11] „Delphi“ yra Borland bendrovės registruotas prekės ženklas.
[12] „Visual Basic“ ir „Visual C++“ yra Microsoft bendrovės registruoti prekės ženklai.
[13] Skliausteliuose pateikiamas PĮ paketo pavadinimas ir versija, kuriam priklauso kompiliatorius.
[14] „Java“ yra Sun Microsystems bendrovės registruotas prekės ženklas.

*Java* ir *Delphi* kompiliatoriai mažiau standartizuoti nei C++. *Java* kalba apskritai neturi standarto, o tik dokumentaciją, kuri iš esmės ir atlieka jo funkcijas. *Delphi* realizuoja Paskalio dialektą, kuris yra kardinaliai išplėstas lyginant su jau pasenusiais Paskalio kalbos standartais ISO 7185 ir ISO 1020.

Principinį šiuolaikinį kompiliatorių sudaro trys pagrindinės dalys: 1) transliatorius, 2) vykdymo meto biblioteka ir 3) saistyklė. O jo veikimą apibūdiną 3 paveiksle pateikta schema. Išsamesnės nagrinėjamų kompiliatorių veikimo schemos pateiktos 3 priede.




*3 pav*. Principinė šiuolaikinio kompiliatoriaus sandara ir veikimo schema

Transliatorius atlieka programavimo kalba (dažniausiai suprantama žmogui) užrašytos programos vertimą į programavimo kalbą skirtą saistyklei. Tai labai panašus darbas kaip ir viso kompiliatorius, todėl praktikoje šie terminai dažnai vartojami sinonimiškai. Šiame darbe šiuos terminus skirsime. Pagrindinė savybė skirianti kompiliatorių nuo transliatoriaus yra tai, kad kompiliatorius į vertimą įkomponuoja jau parengtų programos fragmentų [DGJ05].

Saistyklė naudodama transliatoriaus sugeneruotuose failuose esančią informaciją ir iš anksto parengtus programų fragmentų, laikomų bibliotekose, sugeneruoja galutinius kompiliatoriaus išvedinius (paprastai vykdomuosius failus arba bibliotekas). Dažniausiai jie išreiškiami kompiuteriui suprantamos programavimo kalbos kodais (mašininiais kodais) arba tarpinės kalbos kodais, suprantamais kitiems kompiliatoriams arba virtualioms mašinoms (pavyzdžiui, *.NET* JIT[15] kompiliatoriui).

*Visual C++* ir *Delphi* kompiliatoriuose transliatoriai ir saistyklės yra atskiros, savarankiškos programos. Kituose kompiliatoriuose jie gali būti tarpusavyje integruoti į vieną programą. Mūsų tiriamo *Java* kompiliatoriaus saistyklė nėra aiškiai išreikšta – ja laikytinos Java platformos priemonės skirtos *Java* baitinių programų tikrinimui ir klasių įkėlimui. Java kompiliatoriaus atveju, skirtingai nuo *Visual C++* ir *Delphi* kompiliatorių, kurie atlieka susaistymą kompiliavimo metu, susaistymas atliekamas programos vykdymo metu. Tačiau šis skirtumas tyrimui reikšmės neturi.

Kompiliatoriaus ir programuotojo darbą labai paspartina iš anksto parengtų programų fragmentų saugomų bibliotekose panaudojimas. Transliatoriui jų nebereikia versti, o programuotojui nebereikia jų programuoti. Todėl šiuolaikiniai kompiliatoriai komplektuojami su

[15] Angl. *Just In Time*

įvairios paskirties programų fragmentų bibliotekomis. Svarbiausia iš šių bibliotekų yra vykdymo meto biblioteka (VMB). Ji realizuoja semantinę programavimo kalbos konstrukcijų dalį, standartines funkcijas ir kt. kompiliatoriaus veikimui būtinus elementus. VMB yra kompiliatoriaus dalis be kurios jis negalėtų veikti.

Transliatorius, VMB ir saistyklė sudaro nuoseklią grandinę: transliatorius realizuoja sintaksinę kompiliatoriaus dalį, VMB semantinę, o saistyklė – galutinio kodo generavimo. Todėl kompiliatoriaus internacionalizacijai yra svarbu, kad visos šios trys dalys būtų pakankamai internacionalizuotos. Tačiau svarbiausią vaidmenį kompiliatorių internacionalizacijoje vaidina VMB internacionalizacija. Joje esantys kodo fragmentai kompiliavimo metu yra perkeliami į PĮ, todėl nuo to kiek jie yra internacionalizuoti priklauso tai kiek internacionalizuota bus kompiliatoriumi sukurta PĮ.

Nagrinėjamų kompiliatorių gamintojai truputį skirtingai apibrėžia kas tai yra VMB. *Visual C++* kompiliatoriaus VMB iš esmės atitinka C++ standarto apibrėžiamą standartinę biblioteką ir ji yra aiškiausiai apibrėžta bei dokumentuota (lyginant su kitais nagrinėjamais kompiliatoriais). Java kalbos bibliotekos taip pat gerai dokumentuotos, tačiau naudojama skirtinga terminija. Joje VMB biblioteka įvardijama ne kaip vykdymo meto, o kaip pagrindinė[16]. Nuoseklumo dėlei mes ją vadinsime VMB. *Delphi* VMB yra prasčiau dokumentuota. Remiantis *Delphi* dokumentacija sunku tiksliai nustatyti kurie bibliotekų, platinamų su *Delphi* kompiliatoriumi, moduliai priskiriami VMB, o kurie ne. Todėl laikysime, kad VMB priklauso moduliai esantys *Delphi* paketo pakatalogyje RTL, išskyrus modulius realizuojančius *Windows* OS API. Tokia VMB samprata atitinka *Visual C++*, bei *Java* VMB sampratą, todėl atliekant tyrimą tarp kompiliatorių bus išlaikytas nuoseklumas ir lygybė.

*Delphi 2005* paketą sudaro du skirtingi kompiliatoriai, skirti PĮ kurti *Windows* ir *.NET* platformose. Jų funkcionalumas internacionalizacijos aspektu skiriasi, todėl bus nagrinėjami abu kompiliatoriai.

*Visual C++*, kitaip nei *Delphi*, turi tik vieną kompiliatorių, kuris skirtas kurti tiek *Windows*, tiek .NET platformoms skirtą PĮ. Tačiau šis kompiliatorius neturi atskiros VMB skirtos .NET platformai. .NET atveju *Visual C++* kompiliatorius verčia pirminį tekstą į .NET naudojamas baitines programas ir iš esmės neatlieka jokio susaistymo. Baitinės programos vėliau yra dar kartą kompiliuojamos .NET vykdymo meto priemonėmis, kurioms VMB atstoja .NET aplinkos paslaugos. .NET paslaugų internacionalizacijos nagrinėjimas nėra šio darbo uždavinys, todėl *Visual C++* kompiliatoriaus .NET platformos atžvilgiu nenagrinėsime. Toks nagrinėjimas, taip pat nesiderintu prie kitų kompiliatorių tyrimų.

*Java* kompiliatorius skirtas kurti *Java* platformai skirtą PĮ.

Tarp kompiliatoriaus ir platformos internacionalizacijų egzistuoja ryšys, bet jis nėra esminis. Jei platforma yra internacionalizuota tuomet kompiliatorių lengviau internacionalizuoti panaudojant internacionalizuotas platformos paslaugas. Tačiau platformos neinternacionalizuotumas nėra kliūtis kompiliatoriaus internacionalizavimui. Tokiu atveju neinternacionalizuotas platformos paslaugas, derėtų pakeisti internacionalizuotomis paslaugomis, kurios galėtų būti realizuotos atskirai arba paties kompiliatoriaus.

## 3.2. Tyrimo metodas

Remiantis pirmajame skyriuje pateikta internacionalizacijos samprata, PĮ internacionalizuotumo lygio vertinimo metodų apžvalga bei antrajame skyriuje pateikta

[16] Angl. *Base libraries*

kompiliatorių samprata, buvo sudarytas metodas kompiliatorių internacionalizacijos lygiui įvertinti. Jo pagrindą sudaro klausimynas.

Tyrimas atliekamas analizuojant kompiliatorių dokumentacijas ir tikrinant jų savybes empiriškai – naudojant testus. Dokumentacijos ne visada pakankamai tiksliai aprašo nagrinėjamus objektus, todėl papildomai tikrinant kompiliatorių savybes empiriškai užtikrinama, kad tyrimas bus atliktas kokybiškai.

Naudingos papildomos informacijos gali būti gauta nagrinėjant kompiliatorių pirminį tekstą. Tokią galimybė egzistuoja, nes visų kompiliatorių pirminis tekstas yra iš dalies (paprastai atviras tik VMB pirminis tekstas), o *Java* visiškai atviras.

Klausimyną sudarančių reikalavimų aibę žymėsime $\{R_i\}$. Jie apima kompiliatorių internacionalizacijos aspektus, bet nėra susiję su konkrečiomis tiriamų kompiliatorių galimybėmis. Be to reikalavimai sudaryti taip, kad tyrimui neturėtų įtakos platforma, kurioje veikia kompiliatoriai.

Toliau pateikiamas metodą sudarantis klausimynas su trumpais paaiškinimais. Reikalavimus papildančių testų pavyzdžiai pateikti 4 priede.

1. Duomenų kodavimas[17]
   Duomenų kodavimas plačiau paaiškintas 4.3 skyrelyje. Kompiliatorius išorinių duomenų (pirminio teksto, parinkčių ir kt. failų) kodavimui turi naudoti Unikodą. Duomenys neturi būti sugadinami jų apdorojimo metu dėl klaidingai atliekamo perkodavimo ar kitokių klaidingai atliekamų veiksmų susijusių su PĮ internacionalizacija ir turi būti teisingai išvedami.
2. Išteklių atskyrimas
   Išteklių atskyrimas plačiau paaiškintas 4.4 skyrelyje. Kompiliatoriaus naudojami tekstiniai ištekliai turi būti atskirti nuo veiksmų dalies. Vartojama terminija ir stilius turi atitikti 4.1.2 skyrelyje išdėstytus reikalavimus. Turi būti numatyta galimybė derinti žodžių gramatines formas, leidžiama į verčiamą tekstą įterpti ženklų kodus. Vienodoms eilutėms turi būti priskirti skirtingi identifikatoriai. Eilučių įterpimui ir jungimui naudojamos formatavimo funkcijos. Turi būti pateiktas vartojamų terminų žodynas. Eilutės turi turėti komentarus paaiškinančius jų vartojimo kontekstą. VMB naudojami pranešimai kompiliavimo metu turi būti atskiriami nuo veiksmų dalies tam, kad juos būtų galima lokalizuoti po to kai jie bus įkompiliuoti į išvedinį.
   2.1. Tekstinių išteklių atskyrimo realizacija
      Kompiliatorius turi realizuoti tekstinių išteklių atskyrimo galimybę. Turi būti leidžiama į tekstą įterpti valdymo ženklų kodus, numatyta galimybė pranešimams priskirti komentarus paaiškinančius jų vartojimo kontekstą. Vienodoms eilutėms turi būti priskiriami skirtingi identifikatoriai.
   2.2. Kitų išteklių atskyrimas:
      Kompiliatoriaus naudojami netekstiniai ištekliai turi būti atskirti nuo veiksmų dalies. Netekstiniais ištekliais laikoma: sparčiųjų klavišų kombinacijos, paveikslai ir piktogramos, žymekliai, garso ir vaizdo failai, dialogų šablonai (langų geometrija, spalvos ir kt. parametrai), meniu šablonai, šriftai, šriftų aprašai (pavadinimai, stiliai, dydis) ir kiti ištekliai (animuotos piktogramos ir žymekliai, hipertekstas, lentelės ir t.t.).
      Pažymėtina, kad netgi tekstinės veiksenos kompiliatoriai taip pat gali naudoti netekstinius išteklius. Pavyzdžiui, šriftus, kurie būtų labiau tinkami programos tekstui vaizduoti, nei įprasti platformos šriftai, nes daugumoje platformų fiksuoto pločio šriftų pasirinkimas yra gana menkas.

[17] Čia ir visur kitur šiame darbe turima omenyje tekstiniai duomenys, tačiau dėl paprastumo sutrumpinama.

2.3. Kitų išteklių atskyrimo realizacija:

Kompiliatorius turi realizuoti netekstinių išteklių atskyrimo nuo veiksmų dalies galimybes. Netekstiniais ištekliais laikomi 2.3 punkte išvardinti elementai.

3. Įvedimas

Įvedimas plačiau paaiškinamas 4.7.2 skyrelyje.

3.1. Įvedamų duomenų kodavimas

Kompiliatoriaus realizuota įvedimo paslauga turi leisti įvesti Unikodu koduotą tekstą. Taip pat turi būti numatyta perkodavimo galimybė jei duomenys įvedami iš Unikodo nepalaikančio srauto. Perkodavimas turi būti atliekamas teisingai, pagal nustatytoje lokalėje galiojančias nuostatas.

3.2. Įvedimo paslaugos realizacija

Kompiliatorius turi realizuoti įvedimo paslaugą palaikančią įvairius įvedimo metodus. Be paprastų įvedimo metodų turi būti palaikomi naudojantys trečią lygį, kompleksiniai bei dvikrypčio teksto įvedimo metodai. Taip pat turi būti numatyta galimybė įvesti ženklus surenkant jų kodus paspaudus alternatyvos klavišą.

4. Išvedimas

Išvedimas plačiau paaiškinamas 4.7.2 skyrelyje.

4.1. Išvedamų duomenų kodavimas

Kompiliatoriaus realizuota teksto išvedimo paslauga turi leisti išvesti Unikodu koduotą tekstą. Taip pat turi būti numatyta perkodavimo galimybė, jei duomenys išvedami į Unikodo nepalaikantį srautą. Perkodavimas turi būti atliekamas teisingai, pagal nustatytoje lokalėje galiojančias nuostatas.

4.2. Teksto vaizdavimo paslauga

Kompiliatorius turi realizuoti teksto vaizdavimo paslaugą. Ji turi realizuoti tiek įprastų, tiek kompleksinių, dvikrypčių bei CJK raštų vaizdavimą, teisingai vaizduoti kombinacinių ženklų sekas.

Paslauga turi turėti išplėtimo galimybę tiek VMB, tiek jau sukompiliuotų programų lygyje. Pavyzdžiui, to gali prireikti lokalizuojamą programą papildant nestandartinių šriftų (pavyzdžiui, naudojančių kirčiuotas raides) vaizdavimo galimybėmis.

5. Teksto dorojimas

Teksto dorojimas plačiau paaiškinamas 4.7.3 skyrelyje.

5.1. Kodavimas

Kompiliatoriaus turi realizuoti Unikodo perkodavimo į kitas plačiai naudojamas koduotes ir atvirkščiai paslaugą.

5.2. Ženklų savybės

Kompiliatorius turi realizuoti Unikodo standartą atitinkančią ženklų savybių nustatymo paslaugą.

5.3. Raidžių lygio keitimas

Kompiliatorius turi realizuoti Unikodo standartą ir pasirinktos lokalės nuostatas atitinkančią raidžių lygio keitimo paslaugą.

5.4. Teksto skaidymas

Kompiliatorius turi realizuoti Unikodo standartą ir lokalės nuostatas atitinkančią teksto skaidymo paslaugą. Paslaugos uždaviniai – teksto skaidymas į grafemas, žodžius ir sakinius.

5.5. Teksto normalizavimas

Kompiliatorius turi realizuoti Unikodo standartą atitinkančią teksto normalizavimo paslaugą.

Pažymėtina, kad teksto normalizavimo paslauga yra naudojama kitų paslaugų (pavyzdžiui, rikiavimo) ir turi įtakos jų veikimui.

5.6. Dvikrypčio teksto transformavimas

Kompiliatorius turi realizuoti Unikodo standartą atitinkančią dvikrypčio teksto transformavimo paslaugą.

6. Kultūriniai elementai

Kultūriniai elementai plačiau paaiškinami 4.7.4 skyrelyje.

6.1. Kultūrinių elementų įkėliklis

Kompiliatorius turi realizuoti kultūriniai elementų įkėlimo paslaugą.

6.2. Datos ir laikas

Kompiliatorius turi realizuoti datų ir laiko paslaugą. Paslauga turi palaikyti įvairius pasaulyje naudojamus kalendorius bei laiko juostas.

6.3. Formatavimas ir analizė

Kompiliatorius turi realizuoti formatavimo ir analizės paslaugą. Paslauga turi atlikti datų ir laiko, skaičių, piniginių sumų formatavimą ir analizę. Tai turi būti atliekama naudojant pasirinktoje lokalėje galiojančius šių elementų formatus.

6.4. Pranešimų formatavimas

Kompiliatorius turi realizuoti pranešimų formatavimo paslaugą. Ji skirta parametrizuotam eilučių įterpimui ir jungimui. Parametrų pagalba įterpiami datos ir laikas, skaičiai, piniginės sumos turi būti automatiškai formatuojamos naudojant formatavimo ir analizės paslaugą. Turi būti realizuota žodžių daugiskaitinių formų derinimo galimybė.

6.5. Rikiavimas, palyginimas ir paieška

Kompiliatorius turi realizuoti rikiavimo, palyginimo ir paieškos paslaugas. Paslaugos skirtos atlikti tekstinių sąrašų rikiavimui, teksto eilučių palyginimui, fragmento paieškai tekste arba elemento paieškai tekstiniame sąraše. Šios paslaugos tarpusavyje susiję, jos turi atitikti Unikodo standartą ir lokalės nuostatas. Veiksmai turi būti atliekami atsižvelgiant arba neatsižvelgiant į raidžių lygį.

6.6. Kompiliatoriaus realizacija

Kompiliatorius turi doroti informaciją atsižvelgdamas į nustatytoje lokalėje galiojančias kultūrines nuostatas. T.y. teisingai formatuoti išvedamą informaciją, teisingai įterpdamas datas ir laiką, skaičius. Teisingai rikiuoti, palyginti arba ieškoti.

7. Leksikos elementai

Leksikos elementai plačiau paaiškinami 4.5 skyrelyje.

7.1. Vardai

Kompiliatorius turi realizuoti vardų sintaksę suderinamą su Unikodo standartu. Turi būti teisingai realizuota vardų semantika.

7.2. Eilutės

Kompiliatorius turi realizuoti Unikodo eilučių duomenų tipą.

7.3. Baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai.

Baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai turi būti atskirti nuo veiksmų dalies tam, kad juos būtų galima lokalizuoti. Turi būti galimybė operacijų ženklams priskirti sinonimus.

7.4. Skaičiai

Turi būti galimybė skaičių užrašymui naudoti įvairių raštų (ne tik lotyniško) dešimtainius skaičius.

7.5. Skyryba

Turi būti numatyta skyrybos ženklų lokalizavimo galimybė.

8. Failų sistemos paslaugos

Kompiliatorius turi naudoti ir realizuoti su Unikodu suderintas failų sistemos paslaugas.

Tyrimas buvo atliekamas dviem etapais. Pirmojo etapo metu į reikalavimus $R_i$ buvo atsakoma vienu iš galimų atsakymo variantų: „Taip“, „Ne“, „Nepakankamai“ ir „Netaikoma“, reiškiančių elemento atitikimą suformuluotiems reikalavimams. Tokiu būdu kompiliatorių

internacionalizuotumo lygis įvertintas kokybiškai. Tyrimo rezultatai pateikti 5 priede ir apžvelgti tolesniame skyrelyje.

Antrojo etapo metu įvertinti tik su VMB internacionalizuotumo reikalavimai. Neatitikimas reikalavimui įvertinamas ekspertiniu būdu, pirminio teksto eilučių skaičiumi, kurių gali prireikti šalinant šiuos trūkumus internacionalizuotos programinės įrangos kūrimo metu. Ekspertų apklausa atliekama ir gauti rezultatai apibendrinami taikant Delphi ekspertinių apklausų metodą [RW01]. Taip gaunama įverčių aibė *{$Q_i$}*. Bendrą kompiliatoriaus internacionalizuotumo lygį nusako įvertis $Q = \sum_{\forall i} Q_i$ .

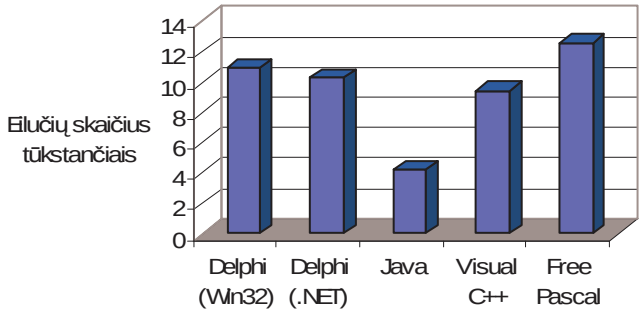
Atlikus tyrimą gauti rezultatai pateikti 1 lentelėje.

## 3.3. Tyrimo rezultatų apžvalga

*1 lentelė*. Kompiliatorių internacionalizuotumo įvertinimas

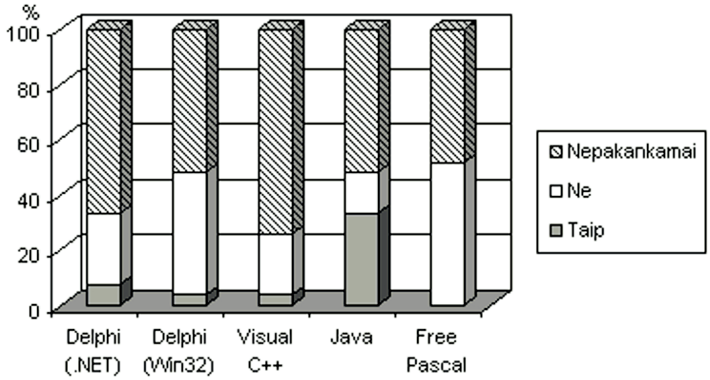
| | | **Įverčiai $Q_i$** | | | | |
|---|---|---|---|---|---|---|
| **i** | **Reikalavimai $R_i$** | ***Delphi (Win32)*** | ***Delphi (.NET)*** | ***Java*** | ***Visual C*** | ***Free Pascal*** |
| | Internacionalizuotos duomenų struktūros | | | | | |
| 1.1. | Internacionalizuotos duomenų struktūros | 100 | 0 | 0 | 100 | 200 |
| | Kompiliatoriaus ištekliai | | | | | |
| 2.1. | Tekstinių išteklių atskyrimas nuo veiksmų dalies | 0 | 0 | 0 | 0 | 540 |
| | Išteklių atskyrimo paslauga | | | | | |
| 3.1. | Tekstinių išteklių atskyrimas nuo veiksmų dalies | 0 | 0 | 0 | 0 | 0 |
| 3.2. | Žodžių gramatinių formų derinimas | 400 | 400 | 200 | 400 | 200 |
| 3.3. | Eilučių identifikavimas nevienareikšmiais identifikatoriais | 0 | 0 | 0 | 0 | 0 |
| 3.4. | Eilučių įterpimas viena į kitą arba jungimas | 0 | 0 | 0 | 0 | 0 |
| 3.5. | Kitų išteklių atskyrimas nuo veiksmų dalies | 0 | 0 | 500 | 0 | 500 |
| | Įvedimo paslauga | | | | | |
| 4.1. | Kodavimas | 200 | 200 | 50 | 200 | 300 |
| 4.2. | Įvedimo metodai turintys trečią lygį arba antrą ženklų grupę | 0 | 0 | 0 | 0 | 0 |
| 4.3. | CJK įvedimo metodai | 500 | 500 | 500 | 500 | 500 |
| | Išvedimo paslauga | | | | | |
| 5.1. | Kodavimas | 200 | 100 | 50 | 100 | 300 |
| 5.2. | Kompleksinių ir CJK raštų teksto išvedimas | 2500 | 2500 | 2500 | 2500 | 2500 |
| | Teksto dorojimo paslauga | | | | | |
| 6.1. | Kodavimas | 300 | 300 | 0 | 300 | 300 |
| 6.2. | Ženklų savybės | 200 | 200 | 0 | 0 | 200 |
| 6.3. | Raidžių lygio keitimas | 150 | 150 | 50 | 50 | 150 |
| 6.4. | Teksto skaidymas | 800 | 800 | 0 | 800 | 800 |
| 6.5. | Teksto normalizavimas | 600 | 600 | 50 | 600 | 600 |
| 6.6. | Dvikrypčio teksto transformavimas | 700 | 700 | 0 | 700 | 700 |
| | Lokalė | | | | | |
| 7.1. | Lokalės duomenų įkėlimas | 800 | 600 | 0 | 300 | 800 |
| 7.2. | Kalendorius, data ir laikas | 400 | 300 | 0 | 300 | 600 |
| 7.3. | Formatavimas ir analizavimas | 2500 | 2000 | 50 | 2000 | 2500 |
| 7.4. | Rikiavimas, palyginimas | 300 | 300 | 0 | 150 | 300 |
| 7.5. | Paieška | 300 | 300 | 300 | 300 | 300 |
| | Platformos paslaugos | | | | | |
| 8.1. | Failų sistemos paslaugos | 0 | 50 | 0 | 50 | 50 |
| 8.2. | Kitos paslaugos | 0 | 250 | 0 | 100 | 250 |

5 paveiksle pateiktas apibendrintas kompiliatorių VMB internacionalizuotumo lygis. Iš diagramos matyti jog Delphi, Visual C++ ir *Free Pascal* kompiliatorių rezultatai yra gana panašūs. Jų vykdymo meto bibliotekoms internacionalizuoti papildomai prireiktų ~10 000–12 000 pirminio tekto eilučių. Tai yra gana žemas internacionalizacijos lygis. Pagrindiniai šių kompiliatorių trūkumai susiję su įvedimo ir išvedimo, teksto dorojimo ir lokalės paslaugų realizacija. Kiek geresnis yra Java kompiliatoriaus rezultatas, tačiau ir jo VMB internacionalizavimui papildomai reikėtų ~4 000 pirminio teksto eilučių.



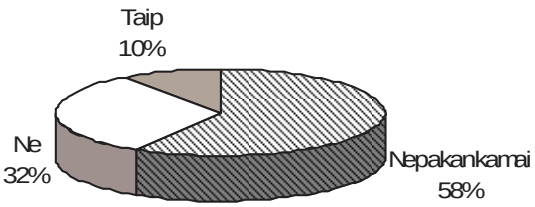

*4 pav*. Apibendrintas kompiliatorių internacionalizuotumo lygis

Atkreiptinas dėmesys į tai, kad visų kompiliatorių internacionalizuotumo lygis yra labai žemas (6 pav.). Tik *Java* kompiliatorius pilnai tenkina 30 % reikalavimų (grafike ši stulpelio dalis nuspalvinta pilkai). Kiti kompiliatoriai netenkina ir 10 %.




*5 pav*. Kompiliatorių internacionalizuotumo lygis

7 paveiksle pateiktas apskaičiuotas kompiliatorių internacionalizuotumo lygio vidurkis. Derėtų atkreipti dėmesį į tai, kad tik 12 % reikalavimų yra patenkinti pilnai, o didžiausia dalis – 61 % tenkinami tik iš dalies.




*6 pav*. Kompiliatorių internacionalizuotumo lygio vidurkis

Toliau gauti rezultatai apžvelgti šiek tiek detaliau, nagrinėjant pastebėtų trūkumų priežastis ir palyginant kompiliatorius tarpusavyje.

### 3.3.1. Duomenų kodavimas

*2 lentelė*. Duomenų kodavimas

| **Duomenys** | ***Delphi (.NET, Win32), Free Pascal*** | ***Visual C++, Java*** |
|---|---|---|
| Pirminio teksto failai | UTF-8 ir pagrindinė OS 8 bitų koduotė | UTF-8, UTF-16, įvairios 8 bitų koduotės |
| Parinkčių failai | Pagrindinė OS 8 bitų koduotė | Pagrindinė OS 8 bitų koduotė |
| Išvedami duomenys | Pagrindinė OS 8 bitų koduotė | Pagrindinė OS 8 bitų koduotė |

Visi kompiliatoriai leidžia pirminio teksto kodavimui naudoti Unikodą. Pirminio teksto kodavimas yra vienas iš svarbiausių kompiliatoriaus internacionalizacijos lygio rodiklių, nes netinkamas pirminio teksto kodavimas užkerta kelią daugelio kitų kompiliatoriaus elementų internacionalizavimui.

Visi kompiliatoriai leidžia naudoti tik pagrindinę OS 8 bitų koduotę parinkčių failams koduoti. To nepakanka, nes kai kurioms parinktims (pavyzdžiui, toms į kurias gali būti įtraukti failų vardai) koduoti būtina naudoti Unikodo kodavimą.

Visi kompiliatoriai neteisingai koduoja į *Windows* konsolę (tekstinę sąsają) išvedamus duomenis. Taip atsitinka todėl, kad *Windows* konsolėje dažniausia naudojama numatytoji, DOS koduotė. Todėl išvedama informacija turi būti perkoduojama arba pakeičiama konsolės koduotė (jei OS tai leidžia). To neatlieka nei vienas kompiliatorius.

### 3.3.2. Išteklių atskyrimas

Visų kompiliatorių gamintojai yra sudarę galimybes lokalizuoti kompiliatorių pranešimus. Nors jų lokalizavimas yra pačių gamintojų prerogatyva ir oficialios informacijos apie jų lokalizavimo galimybes jie nepateikia, tačiau ištyrus *Visual C++* ir *Delphi* kompiliatorius įprastomis lokalizavimui skirtomis priemonėmis buvo nustatyta, kad jų lokalizuojami ištekliai yra įkompiliuoti į vykdomųjų failų arba su jais susiejamų dinaminių bibliotekų išteklių sekcijose, taikant *.RC* metodą. *Java* kompiliatorius naudoja *ResourceBundles* metodą. Šiems metodams būdingi trūkumai ir pranašumai pateikti 4.6 skyrelyje.

*3 lentelė*. Išteklių atskyrimo metodai

| ***Delphi (.NET)*** | ***Delphi (Win32)*** | ***Visual C++*** | ***Java*** | ***Free Pascal*** |
|---|---|---|---|---|
| *.RESX* | *.RC* | *.RC, .RESX* | *ResourceBundles* | *GetText* |

Kompiliatorių realizuojami išteklių atskyrimo metodai priklauso nuo to, kuriai platformai skirta jais kuriama PĮ: *Windows* platformai naudojamas *.RC* metodas, *.NET – .RESX*, *Java – ResourceBundles*. Taip einama lengviausiu keliu, nes visi šie metodai turi trūkumų (žr. 4.6 skyrelį). Pažymėtina, kad platforma neriboja to koks atskyrimo metodas gali taikomas jai skirtose programose, ji tik suteikia paslaugas, kuriomis naudojantis atitinkami metodai lengviau realizuojami.

Išteklių atskyrimo metodų realizacija priklauso ir nuo kompiliatoriaus. Pavyzdžiui, nors *Visual C++* ir *Delphi* naudoja tuos pačius metodus išteklių atskyrimui, tačiau *Delphi* juos realizuoja geriau. *Delphi* išteklius kompiliavimo metu automatiškai susieja su programavimo kalbos konstrukcijomis ir automatiškai juos įkelia. *Visual C++* (taip pat ir *Java*) atveju išteklių įkėlimą turi aprašyti programuotojas, o tai sukelia papildomų sunkumų.

Daugumą su išteklių įkėlimu susijusių problemų galima išspręsti į programavimo kalbos realizaciją įtraukiant papildomas konstrukcijas skirtas išteklių aprašams, kurios leistų išteklių įkėlimą labiau automatizuoti.

### 3.3.3. Įvedimas ir išvedimas

*4 lentelė.* Įvedimas ir išvedimas

| **Elementas** | ***Delphi (.NET, Win32), Java, Free Pascal*** | ***Visual C++*** |
|---|---|---|
| Įvedamų ir išvedamų duomenų kodavimas | Pagrindinė OS 8 bitų koduotė | UTF-16, pagrindinė OS 8 bitų koduotė |
| Įvedimo ir išvedimo realizacija | Palaikomi tik paprasti įvedimo metodai ir alfabetų[18] grupės rašto sistemos | Palaikomi tik paprasti įvedimo metodai ir alfabetų grupės rašto sistemos |

Visi kompiliatoriai įvedimui ir išvedimui naudoja *Windows* OS paslaugas. Nei vienas kompiliatorius ir netgi *.NET* arba *Java* platforma, nerealizuoja savo įvedimo arba išvedimo paslaugos (šiuo atveju jos atlieka tik tarpininko vaidmenį tarp *Windows* OS ir PĮ). Toks realizacijos būdas yra lengviausias, bet ne geriausias, nes šios *Windows* OS paslaugos nėra pakankamai internacionalizuotos. Iš to kyla pagrindiniai trūkumai: palaikomi tik paprasti įvedimo metodai ir alfabetų grupės rašto sistemos.

*Windows* OS įvedimui arba išvedimui yra naudojama tekstinė sąsaja – konsolė. Skirtingose *Windows* versijose konsolės internacionalizuotumo lygis skiriasi. *Win32* šeimai priklausančiose OS leidžiama įvesti arba išvesti tik 8 bitais koduotą tekstą. *WinNT, 2K, XP* šeimos OS palaiko Unikodą, tačiau šis palaikymas yra gana ribotas. Pagrindiniai jo trūkumai:

1) realizuotas tik alfabetų grupės raštų įvedimas ir išvedimas. Konsolėje naudojami tik paprasti įvedimo ir teksto vaizdavimo metodai, paremti fiksuoto pločio šriftais (paprastai naudojamas *Lucida Console* šriftas, kuris apima tik Lotyniškų, Kirilicos ir Graikų raštų ženklus) [Mi03].

2) *Windows* konsolės API yra nepakankamas, kad ją būtų galima internacionalizuoti (pavyzdžiui, įdiegti trūkstamus įvedimo metodus).

Įvedimo arba išvedimo internacionalizavimo problemą derėtų spręsti pakeičiant *Windows* OS konsolę į naują tekstinę sąsają. Panašiai tekstines sąsajas realizuoja dauguma telneto programų. Tokia sąsaja galėtų būti internacionalizacijos karkaso (žr. 3 skyrelį) dalimi. Tais atvejais, kai programa vykdoma, kitoje tekstinėje terpėje (pavyzdžiui, *Windows* konsolėje, komandų interpretatoriuje (pvz. *4NT*), telneto programoje ir pan.) ji turėtų atlikti tarpininko vaidmenį tarp tos terpės ir programos.

Visi kompiliatoriai turi trūkumų susijusių su įvedamo ir išvedamo teksto kodavimu. *Delphi* ir *Java* kompiliatoriai realizuoja tik 8 bitais koduoto teksto įvedimą ir išvedimą, nepaisant to, kad įvedimo arba išvedimo funkcijų parametrai yra koduojami Unikodu (šiuo atveju tekstas yra perkoduojamas į 8 bitų koduotę net ir tais atvejais kai srautas iš kurio jis įvedamas arba išvedamas leidžia naudoti Unikodą). Pavyzdžiui, *Visual C++* kompiliatoriaus funkcijos *wscanf*, *wprintf*, *_getws*, *_putws* papildomai dar ir klaidina programuotoją, nes jų pavadinime esanti raidė „w“ žymi suderinamumą su Unikodu kas yra netiesa. Dar didesnis *Visual C++* kompiliatoriaus trūkumas tas, kad dalis funkcijų turinčių Unikodo eilučių parametrus išvedant jį paprasčiausiai nukerpa pradedant nuo pirmo ne ASCII kodų lentelės ženklo.

[18] Žr. 4.1.1 skyrelį

### 3.3.4. Teksto dorojimas

*5 lentelė*. Teksto dorojimas

| **Paslauga** | ***Delphi (.NET, Win32), Free Pascal*** | ***Visual C++*** | ***Java*** |
|---|---|---|---|
| Kodavimas | UTF-16, UTF-8, pagrindinė OS 8 bitų koduotė | UTF-16, UTF-8, pagrindinė OS 8 bitų koduotė | Įvairios koduotės |
| Ženklų savybės | Nerealizuota | Tik pagrindinės ženklų savybės | Teisingai |
| Raidžių lygio keitimas | Neatsižvelgiama į lokalės nuostatas | Neatsižvelgiama į lokalės nuostatas | Teisingai |
| Teksto skaidymas | Nerealizuota | Neatsižvelgiama į kombinacines sekas | Teisingai |
| Teksto normalizavimas | Nerealizuota | Nerealizuota | Nerealizuota |
| Dvikrypčio teksto transformavimas | Nerealizuota | Nerealizuota | Teisingai |

Tik *Java* kompiliatorius realizuoja teisingai realizuoja daugumą teksto dorojimo paslaugų. Visual C++ kompiliatorius realizuoja daugumą paslaugų, tačiau jos turi trūkumų. Nei vienas kompiliatorius nerealizuoja teksto normalizavimo paslaugos, nors ji yra labai svarbi. Nuo jos priklausomos teksto rikiavimo, palyginimo ir paieškos paslaugos.

### 3.3.5. Kultūriniai elementai

*6 lentelė*. Kultūriniai elementai

| **Paslauga** | ***Delphi (.NET), Visual C++*** | ***Delphi (Win32), Free Pascal*** | ***Java*** |
|---|---|---|---|
| Įkėlimas | Įkeliami tik pagrindiniai kultūriniai elementai. Remiasi OS lokalių aprašais. | Nerealizuota | Teisingai |
| Datos ir laikas | Grigaliaus ir Julijaus kalendoriai | Grigaliaus ir Julijaus kalendoriai | Grigaliaus kalendorius. Numatyta galimybė įdiegti kitus kalendorius |
| Formatavimas ir analizė | Numatyti tik pagrindiniai formatavimo elementai, be to formatavimas ne visada atliekamas teisingai | Nerealizuota | Teisingai |
| Pranešimų formatavimas | Nenumatyta gramatinių formų kaitymo galimybė | Nenumatyta gramatinių formų kaitymo galimybė | Nenumatyta gramatinių formų kaitymo galimybė |
| Rikiavimas, palyginimas ir paieška | Neatliekamas teksto normalizavimas | Nerealizuota | Teisingai |

Tik *Java* kompiliatorius teisingai realizuoja daugumą kultūrinių elementų paslaugų. Skirtingai nei kiti kompiliatoriai jis naudoja ne OS lokalės aprašus (kuriuos sudaro gana siaura kultūrinių elementų aibė), o *Java* platformos aprašus kurie yra atviri ir reikalui esant gali būti išplėsti papildomais elementais. Galimybę išplėsti kultūrinių elementų aprašus realizuoja tik *Java* kompiliatorius. Pažymėtina, kad nors *Java* kompiliatorius nerealizuoja teksto

normalizavimo paslaugos, tačiau rikiavimo, palyginimo arba paieškos paslauga turi kombinacinių sekų dekompozicijos galimybę.

*Visual C++* ir *Delphi .NET* kompiliatorių trūkumai yra bendri, nes paslaugų realizacijos pagrindą sudaro nepakankamai internacionalizuotos *Windows* OS paslaugos. Sprendžiant šią problemą tikslinga būtų realizuoti šias paslaugas iš naujo (žr. 4.2 ir 4.4.4 skyrelius).

Daugiausia trūkumų turi *Delphi* Win32 skirtas kompiliatorius, nors objektyvių priežasčių paaiškinančių šių trūkumų priežastis ir jo skirtumą nuo *.NET* aplinkai skirto kompiliatoriaus nėra.

### 3.3.6. Leksikos elementai

*7 lentelė*. Leksikos elementai

| **Elementai** | ***Delphi .NET*** | ***Delphi Win32, Visual C++, Free Pascal*** | ***Java*** |
|---|---|---|---|
| Vardai | Neatitinka Pascal kalbos vardų sudarymo taisyklių | Leidžiama naudoti tik ASCII ženklus | Teisingai |
| Eilutės | Teisingai | Neteisingai konvertuojamos iš vieno tipo į kitą | Teisingai |
| Baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai, skaičiai, skyryba | Neleidžiama lokalizuoti | Neleidžiama lokalizuoti | Neleidžiama lokalizuoti |

Kompiliatoriai skirtingai realizuoja vardų sintaksę. Nors 1999 atnaujintas C++ kalbos standartas jau apibrėžia Unikodo ženklų naudojimą varduose, tačiau tirtas *Visual C++* kompiliatorius dar neleidžia jų naudoti. Java kompiliatoriaus realizuojama vardų sintaksė iš esmės suderinama su Unikodo standartu, tik neatliekamas kombinacinių sekų normalizavimas, bet tai nėra labai reikšmingas trūkumas ir jis yra pateiktas kompiliatoriaus dokumentacijoje. *Delphi .NET* kompiliatorius taip pat leidžia varduose naudoti Unikodo ženklus, tačiau jų sintaksės nesiderina su Pascal kalbos taisyklėmis, nes nėra atsižvelgiama į ženklų savybes (pavyzdžiui, vardai gali prasidėti lietuviškomis kabutėmis ar arabiško rašto skaitmenimis).

Visi kompiliatoriai realizuoja Unikodo eilutes. *Delphi .NET* ir *Java* kompiliatoriuose pagrindinis duomenų eilučių duomenų tipas. *Visual C++* taip pat leidžia jį nustatyti kaip pagrindinį naudojant parinktis arba direktyvas.

*Delphi* Win32 ir *Visual C++* kompiliatoriai klaidingai tarpusavyje konvertuoja (pavyzdžiui, priskyrimo arba duomenų tipo keitimo metu) Unikodo ir 8 bitų eilutes. Tai svarbus trūkumas galintis tapti informacijos praradimo priežastimi.

Nei vienas kompiliatorius nerealizuoja bazinių žodžių, direktyvų vardų, modifikatorių, operacijų ženklų, skaičių arba skyrybos lokalizavimo galimybių.

### 3.3.7. Failų sistemos paslaugos

Visi kompiliatoriai naudoja nesuderintą su Unikodu failų sistemos API ir neleidžia, naudoti failų, kurių vardų kodavimui naudojamas Unikodas.

Teisingai failų sistemos paslaugas realizuoja tik *Java* ir *Delphi .NET* platformai skirti kompiliatoriai. Kiti kompiliatoriai realizuoja tik 8 bitais pagristas failų sistemos paslaugas.

## 3.4. Išvados

Nustatyta, kad:

1. Visų nagrinėtų kompiliatorių internacionalizuotumo lygis yra žemas.
2. Visi kompiliatoriai pirminio teksto kodavimui leidžia naudoti Unikodą, bet jo palaikymas yra ribotas.
3. Visi kompiliatoriai turi trūkumų susijusių su teksto įvedimo ir išvedimo paslaugos realizacija. Pagrindiniai šios paslaugos trūkumai susiję su naudojamu teksto kodavimu 8 bitais.
4. Visi kompiliatoriai naudoja skirtingus išteklių atskyrimo metodus. Visi šie metodai turi ties savų pranašumų, tiek trūkumų.
5. Visų kompiliatorių realizuotos teksto apdorojimo paslaugos yra nesuderintos arba nepakankamai suderintos (tik *Java* kompiliatoriaus) Unikodo standartu.
6. Nei vienas iš kompiliatorių, išskyrus *Java*, nerealizuoja arba nerealizuoja pakankamai gerai kultūrinių nuostatų įkėlimo ir su jomis susijusių paslaugų.
7. Tik *Java* kompiliatorius leidžia pirminiame tekste naudoti vardus suderinamus su Unikodo standartu ir atitinkančius programavimo kalbos taisykles.
8. Nei vienas kompiliatorius neleidžia lokalizuoti programavimo kalbos leksikos.
9. Visi kompiliatoriai naudoja tik 8 bitais koduotus failų vardus ir neteisingai arba nepakankamai gerai realizuoja failų sistemos paslaugas.

# IV. KOMPILIATORIŲ INTERNACIONALIZAVIMO METODAS

## 4.1. Pagrindiniai kompiliatorių internacionalizavimo uždaviniai

Pagrindiniai uždaviniai keliami internacionalizuojant bendros paskirties PĮ yra:

1. Išplėsti PĮ branduolį atliekantį veiksmus tam, kad jis būtų suderinamas su įvairiomis lokalėmis.
2. Atskirti nuo veiksmus atliekančio PĮ branduolio lokalizuotinus išteklius (PĮ sąsajos ir pranešimų tekstus, dialogų šablonus, paveikslėlius, piktogramas ir kt.).

Internacionalizuojant kompiliatorius siekiama, kad jais kuriama PĮ būtų internacionalizuota, todėl jų internacionalizavimo uždaviniai išplaukia iš bendrosios paskirties PĮ internacionalizavimo uždavinių:

1. Įdiegti vykdymo meto bibliotekoje PĮ internacionalizavimui skirtas paslaugas.

   PĮ internacionalizavimui skirtos paslaugos plačiau aprašytos 4.6 skyrelyje.

2. Išplėsti vykdymo meto biblioteką tam, kad joje esantys kodo fragmentai būtų suderinami su įvairiomis lokalėmis.

   Suderinamumas su įvairiomis lokalėmis gali būti pasiektas įdiegiant Unikodo kodavimo metodą (žr. 4.3 skyrelį) ir vykdymo meto bibliotekos išplėtimams naudojant PĮ internacionalizavimui skirtas paslaugas (žr. 4.6 skyrelį), o pats lokalių mechanizmas plačiau aprašytas 4.2 ir 4.7.1 skyreliuose.

3. Įdiegti vykdymo meto bibliotekoje išteklių atskyrimo paslaugą.

   Išteklių atskyrimo paslauga plačiau aprašyta 4.4 ir 4.7.6 skyreliuose.

4. Išplėsti kompiliatorių tam, kad jis būtų suderinamas su įvairiomis lokalėmis.

   Kompiliatorius kaip ir bendros paskirties PĮ turi būti internacionalizuotas. Tačiau kompiliatoriaus internacionalizavimo atveju prisideda ir papildomų uždavinių susijusių su programavimo kalbos leksika. Kompiliatorius ne tik turi leisti įvesti programas užrašytas daugiakalbiu tekstu, bet ir teisingai analizuoti jų leksiką, kuri savo ruožtu gali būti susijusi ir su semantika, kuri realizuojama vykdymo meto bibliotekoje ir gali turėti įtakos kompiliatoriumi kuriamos PĮ internacionalizuotumei (pavyzdžiui, Unikodo duomenų tipų realizacija). Plačiau programavimo kalbų leksikos internacionalizavimas apžvelgtas 4.5 skyrelyje.

5. Atskirti kompiliatoriaus naudojamus lokalizuotinus išteklius nuo veiksmų dalies.

   Kompiliatoriaus, kaip ir bendros paskirties PĮ, lokalizuotini ištekliai turi būti atskirti nuo veiksmų dalies. Ypatingai svarbu, kad nuo veiksmų dalies būtų atskirti vykdomo meto bibliotekoje naudojami lokalizuotini ištekliai, nes nuo to priklauso ir tai ar šie ištekliai bus atskirti kompiliatoriumi sukurtoje PĮ. Išteklių atskyrimas plačiau aprašytas 4.4 ir 4.7.6 skyreliuose.

## 4.2. Kultūriniai ir kalbiniai faktoriai PĮ internacionalizacijoje

Internacionalizacijos apimtis galima nustatyti tik išnagrinėjus kokie kultūriniai ir kalbiniai faktoriai įtakoja jos praktiškumą, t.y. į kuriuos kultūrinius ir kalbinius PĮ vartotojo aplikos elementus būtina atsižvelgti ir sudaryti sąlygas jų adaptavimui.

Vieni iš pirmųjų PĮ internacionalizacijai reikšmingus faktorius pabandė suklasifikuoti Mahemoff ir Johnston [MJ98]. Jų sudaryta klasifikacija iš dalies remiasi dvejomis [Ye96] išskirtomis faktorių grupėmis: akivaizdžių ir neakivaizdžių faktorių.

Patys svarbiausi PĮ internacionalizacijai akivaizdūs kultūriniai faktoriai, todėl jie plačiau apžvelgti 4.2 skyrelyje. Jie yra gana aiškūs ir nesunkiai aprašomi formaliu būdu naudojant lokales.

Mažiau svarbūs yra neakivaizdūs faktoriai, tačiau gaminant internacionalizuotą PĮ į juos taip pat būtina atsižvelgti. Šie faktoriai internacionalizavimo metu kelia daugiau problemų nei akivaizdūs, nes juos formaliai apibrėžti neįmanoma. Trumpai juos apžvelgtime.

Mahemoff ir Johnston [MJ98] išskiria šias neakivaizdžių faktorių grupes:

1. Išankstinio nusistatymo. Evers ir Day [ED97] pateikia akivaizdų tokio nusistatymo pavyzdį: ištyrus Indonezijos ir Kinijos naudotojų teikiamus prioritetus konstruojant hipotetinį kompiuterinės programos modelį, paaiškėjo, kad vieni iš jų daugiausia dėmesio skyrė praktiškumui, kiti naudojimo paprastumui. Kitas pavyzdys: dauguma specialistų teigia, kad „Atšaukti“ funkcija yra naudinga, nes leidžia tyrinėti PĮ galimybes, tuo tarpu Ito ir Nakakoji [IN96] teigia, kad Japonai yra nusistatę prieš bandymų ir klaidų metodą, nes laiko kad tai daug laiko reikalaujantis metodas, ir todėl „Atšaukti“ funkcijai jie teikia mažesnę svarbą.
2. Suvokimo. Evers [Ev98] remdamasi Lovgren [Lo94] teigia, kad dauguma galimybių PĮ yra išreikštos metaforiškai: t.y. PĮ objektus sutapatinant su realaus pasaulio objektais. Tačiau šios metaforos skirtingose kultūrose gali būti suprantamos skirtingai. Pavyzdžiui, aplanko piktograma bus nevisai tinkama kultūrose, kuriose dokumentus įprasta laikyti kartoninėse dėžėse [Ma96]. Daugelis žino jog spalvos įvairiose kultūrose taip pat suvokiamos ne vienodai, pavyzdžiui, raudona spalva JAV reiškia pavojų, Kinijoje džiaugsmą. Skirtingose kultūrose taip pat gali skirtis ir pats mastymo bei suvokimo stilius [Ch98].
3. Bendravimo taisyklių. Kūno kalba, veido išraiškos, dialogo palaikymo stiliai skirtingose kultūrose skiriasi. Pavyzdžiui, Edvards [En95] nustatė aštuoniolika kultūrinių grupių, kuriose skiriasi rankų ženklai, naudojami virtualiose aplinkose.
4. Naudojimo terpės. Terpė kurioje dirba PĮ naudotojas skirtingose kultūrose gali skirtis ir tai taip pat turi įtakos PĮ internacionalizacijai. Pavyzdžiui, JAV priimta turėti atskirą uždarą erdvę darbo vietoje, kai tuo tarpu Japonijoje atvirą. Dirbant atviroje erdvėje mažiau reikalingos bendravimo el. laiškais arba žinutėmis priemonės, o garsiniai signalai gali erzinti šalia esančius bendradarbius.

Neakivaizdžių faktorių realizacija yra sudėtingesnė. Ji reikalauja ypatingo pasiruošimo PĮ gamybai, tiriant konkrečioje terpėje galiojančias nuostatas [JCD04]. PĮ kurioje būtų atsižvelgiama į šiuos faktorius gamybos procesas nesibaigia su jos perdavimu lokalizuotojams, nes beveik neįmanoma iš anksto numatyti visų ją įtakojančių kultūrinių faktorių, kurie gali atsiskleisti lokalizuotos versijos testavimo metu. Gamintojas turi būti pasiruošęs pašalinti lokalizuotos versijos testavimo metu atskleistus trūkumus, todėl tarp gamintojo ir lokalizuotojų turi vykti nuolatinis bendradarbiavimas [IS03].

### 4.2.1. Raštai

Raštas – tai sistema ženklų ir taisyklių, kurių pagalba išreiškiama kalba taip, kad ji galėtų būti atkurta kitų, be ją užrašiusiojo įsikišimo. Skirtingose kultūrose kalbai užrašyti naudojami įvairūs ženklai bei taisyklės, todėl egzistuoja didelė raštų įvairovė. Kompiuteryje naudojamų raštų kodus ir pavadinimus apibrėžia standartas ISO 15924 (šiuo metu jis apima 115 raštų

kodus). Toliau sugrupuosime ir trumpai aptarsime pagrindinius raštų skirtumus, kurie turi įtakos jų realizacijai kompiuteryje.

Unikodo standarte [Un03] pateikiama raštų klasifikacija į penkias grupes:

1. Alfabetai.
2. Abdžadai[19] (*abjads*).
3. Abugidai (*abugidas*).
4. Skiemeniniai.
5. Logo-skiemeniniai.

Ši klasifikacija sudaryta remiantis pasaulyje pripažinto raštų eksperto Peter T. Daniels pateikta klasifikacija [DB96]. Ji daugiausia remiasi raštų sandaros savybėmis, t. y. kaip kalba išreiškiama raidėmis. Skirtingose raštuose raidės gali reikšti priebalsius, balsius, skiemenis, žodžius arba jų dalis ir idėjas. Suprantama, kad nuo to labai priklauso ir raštuose naudojamų ženklų kiekis (žymėti priebalsiams jų reikia sąlyginai nedaug, tuo tarpu žymėti žodžiams jų reikia daugybės). Trumpai apžvelgsime šias raštų grupės ir kitus PĮ internacionalizacijai svarbius raštų skirtumus.

**Alfabetai**. Alfabetuose raidės reiškia priebalsius ir balsius. Priebalsiai ir balsiai juose turi tokią pačią vertę. Alfabetų grupei priskiriami lotynų, kirilicos, graikų, gruzinų, runų ir kt. raštai. Iš alfabetų labiausiai yra paplitęs lotyniškasis; jį naudoja tūkstančiai kalbų, taip pat lietuvių.

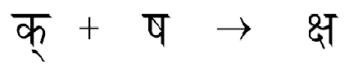
Šios grupės raštų raidės (kaip abdžadų, abugidų ir skiemeninių raštų) daugiau ar mažiau atitinka kalbos garsus. Raštai, kuriose raidės (arba ženklai) atitinka kalbos garsus vadinami fonetiniais. Pavyzdžiui, Tarptautinis Fonetinis Alfabetas [IPA99] [Ha02] buvo sukurtas specialiai tam, kad būtų galima tiksliai užrašyti kalbos garsus. Tačiau daugelyje kalbų ryšys tarp raidžių ir garsų yra netiesioginis, jis apibrėžiamas formaliomis taisyklėmis, bet ir pastarosios gali turėti įvairių išimčių, kaip pavyzdžiui anglų kalboje [Ro00]. Kalbos garsams išreikšti priebalsiais ir balsiais pakanka nedaug raidžių, todėl dauguma šios grupės raštų jų turi apie 30.

**Abdžadai**. Abdžaduose raidės reiškia priebalsius (ir ilgus balsius). Balsiai juose beveik nežymimi arba žymimi tik kaip diakritiniai raidžių ženklai (pavyzdžiui taškai arba kitokie ženklai virš raidžių ir po jomis). Abdžadus daugiausia naudoja vidurio rytų šalys. Geriausiai žinomas abdžado pavyzdys yra arabų raštas. Šiaip grupei taip pat priklauso hebrajų, sirų ir kt. raštai. Šie raštai turi panašų raidžių kiekį kaip ir alfabetai.

**Abugidai**. Agubiduose raidės reiškia priebalsius ir balsius. Tačiau skirtingai nuo alfabetų jie turi ir skiemeninio rašto požymių, nes šių raštų priebalsiai turi prijungtus balsius (daugiausia trumpąjį „a“) (šie balsiai netariami jei po priebalsių eina nepriklausomi balsiai) (dalis autorių juos linkę vadinti angliškų terminu „alphasyllabic“ pabrėždami būtent šį jų bruožą [Br96] [Sp03]). Abugidus daugiausia naudoja pietų Azijos šalys. Šiai grupei priklauso Devanagari, bengalų, tamilų, tibetiečių ir daug kitų raštų. Iš jų plačiausiai naudojamas Devanagari. Jis naudojamas Sanskrito, Hindi ir daugelio kitų kalbų.

Vienas iš aktualiausių šių raštų bruožų yra tai, kad raidžių forma ir netgi jų išsidėstymo tvarka[20] gali kisti priklausomai nuo kitų kontekste esančių raidžių (atskirų raidžių formos pakitimo priklausomai nuo konteksto atvejų galima rasti ir kituose raštuose, tačiau ženklų tvarkos pakitimas yra būdingas tik šiems raštams). Daugeliu atveju raidės gali neatpažįstamai pakeisti savo formą, lyginant su pirminių sujungtų raidžių forma. 9 paveiksle pateikiamas tokio raidžių junginio pavyzdys. Šių raštų realizacijos kompiuteryje dar vadinami sudėtinėmis.

[19] Abdžado ir abugido pavadinimai sukurti jungiant tiems raštams tipiškų abėcėlių pirmąsias raides.
[20] Tai neturi nieko bendra su dvikrypčių raštų raidžių dėstymo tvarka



*7 pav*. Devanagari raidžių junginio pavyzdys

Kaip ir prieš tai minėtų grupių raštai abugidai taip pat nepasižymi dideliu raidžių kiekiu. Pavyzdžiui, Devanagari raštas jų turi 48. Tačiau jei priskaičiuotume ir visas naujas raides, kurios gaunamos jungiant pirmines raides, jų skaičius gali gerokai išaugti.

**Skiemeniniai raštai**. Skiemeniniuose raštuose raidės[21] reiškia skiemenis. Šios grupės raštus galima išskirti į du pogrupius: 1) paprasti – kuriuose raidžių vaizdavimas neturi ryšio su jos reiškiamo skiemens fonetine sandara (jam priklauso hiragana, katakana, bopomofo, čerokių ir kt. raštai), 2) charakteringieji – kuriuose minėtas ryšys egzistuoja (jam priklauso etiopų, Hangul ir kt. raštai).

Nors šie raštai priklauso tai pačiai grupei, tačiau raidžių skaičiumi jie gali labai skirtis. Pavyzdžiui, Hiragana ir Katakana turi apytikriai[22] po 100 raidžių, etiopų raštas ~300, Hangul ~ 11000 (šio rašto raidės sudarytos iš fonetinių ženkų, kurių yra 24, sekų).

**Logo-skiemeniniai**. Logo-skiemeniniuose raštuose raidės reiškia žodžius arba žodžių dalis. Kiti autoriai rašydami apie šiuos raštus dar naudoja terminus ideografinis, logografinis, piktografinis, labiau paryškindami atskiras šių raštų savybes, tuo tarpu terminas logo-skiemeniniai apima visas šias savybes, todėl yra tikslesnis. Pagrindinis šios grupės atstovas yra Han raštas, kuris yra naudojamas kinų, japonų ir korėjiečių[23,24]. Šis raštas turi gana daug raidžių, pavyzdžiui, kinų *Kangxi* žodyne išleistame 1716 jų pateikta virš 47 000.

Pirminis šio rašto šaltinis yra Kinija. Tik vėliau jis buvo pradėtas naudoti kitose šalyse. Bėgant laikui raštas keitėsi (modernėjo) ir todėl skirtingose šalyse dabar galima rasti šio rašto skirtumų. Be to, moderninant raštą buvo sukurta nauja, supaprastinta jo forma, kuri ženklų vaizdavimo būdu iš esmės skiriasi nuo tradicinės. Dėl šių priežasčių ženkliai padidėjo šiuose raštuose naudojamų raidžių kiekis.

Kiti esminiai raštų skirtumai turintys didelę reikšmę PĮ internacionalizacijai yra [HH97]:

1. Rašymo kryptis.
2. Priklausomybė nuo konteksto.
3. Skyrybos.
4. Apipavidalinimo.
5. Paieškos ir rikiavimo.
6. Eilučių laužymo.
7. Skaitmenų skirtumai.

Trumpai juos apžvelgsime. Daugiau informacijos apie šiuos skirtumus galima rasti Unikodo standarte. Jame nesiekiama, išsamiai aprašyti visus skirtumus. Jie pateikiami daugiausia kaip pavyzdžiai, į kuriuos turi būti atsižvelgta gaminant su šiuo standartu suderinamą PĮ.

**Rašymo kryptis**. Pagal tai raštai yra skirstomi į dvikrypčius ir vienkrypčius. Vienkrypčiai raštai dažniausiai yra rašomi iš kairės į dešinę (pavyzdžiui, lotynų), dvikrypčiai iš

[21] Anglų kalboje dažnai vadinamos terminu „syllables“ lietuviškai reiškiančiu „skiemenys“. Tačiau tokiu atveju terminija tampa ne visai aiški.
[22] Raidžių skaičius rašomas apytikriai todėl, kad dažnai dalis raidžių raštuose būna nenaudojamos (pasenusios), nors formaliai jos egzistuoja, todėl dažnai sunku pasakyti tikslų raidžių skaičių. Be to šiuose pavyzdžiuose jis nėra labai aktualus.
[23] Šiai grupei žymėti anglų kalboje dažnai naudojama CJK santrumpa reiškianti: China, Japan, Korea.
[24] Anksčiau šį raštą naudojo ir vietnamiečiai, tačiau dabar jie yra perėję prie paprastesnės, lotyniškos rašto sistemos.

dešinės į kairę su intarpais rašomais priešinga kryptimi. Pavyzdžiui, arabų rašte arabų raidės yra rašomos iš dešinės į kairę, o skaičiai ir priešingos krypties raštų intarpai iš kairės į dešinę. Prie vienkrypčių raštų taip pat priskiriami raštai kurie yra rašomi vertikaliai (pavyzdžiui, tradicinis kinų).

**Priklausomybė nuo konteksto**. Kaip jau minėta daugumoje abugidų raidės gali keisti savo formą priklausomai nuo konteksto (šalia esančių raidžių). Tai nėra vien tik abugidų bruožas, priklausomai nuo konteksto raidžių forma gali kisti ir kai kuriuose kituose, pavyzdžiui arabų, raštuose [HH97].

**Skyryba**. Skirtinguose raštuose gali būti naudojami skirtingi skyrybos ženklai. Pavyzdžiui, lotyniškame ir daugelyje kitų raštų žodžius priimta skirti tarpais, tuo tarpu tailandiečiai žodžių skyrybos ženklų nenaudoja. Taip pat reikia turėti omenyje, kad skirtingose kultūrose, net ir naudojančiose tą patį raštą gali būti naudojami skirtingi skyrybos ženklai. Pavyzdžiui, anglai citavimui naudojamos kabutės (“”) atrodo kitaip nei lietuviškos („“), nors abi kultūros naudoja tą patį lotynišką raštą.

**Apipavidalinimas**. Skirtingose kultūrose ir raštuose yra nusistovėję įvairios taisyklės kaip turėtų būti apipavidalinamas tekstas. Pavyzdžiui, vienose kultūrose yra priimta svarbius užrašus išskirti kursyvu, kitose – tarpais tarp raidžių arba pastorinant šriftą. Apipavidalinimas taip pat liečia antraščių rašymo, sąrašų ir išnašų žymėjimo bei daugelį kitų nuostatų. Raštų savybės taip pat lemia dalį šių skirtumų, nes tai kas nesunkiai realizuojama viename rašte, gali būti sunkiai realizuojama ar net neįmanoma kitame.

**Paieška ir rikiavimas**. Paieškos ir rikiavimo skirtumus lemia ne tik skirtingose kultūrose nusistovėjusios nuostatos, tačiau ir raštų skirtumai. Pavyzdžiui, alfabetai turi aiškiai apibrėžtas raidžių tvarkos, žodžių ir rašmenų ribas, todėl juos nesunku suskaičiuoti, atlikti jų paiešką arba rikiavimą. Tokias aiškias savybes turi ne visi raštai: jos gali būti nevienareikšmės arba neapibrėžtos. Pavyzdžiui, japonų rašte paprastai naudojami kelių tipų rašmenys: Hiraganos, Katakanos ir Kanji[25] (10 pav.), todėl japonų paieškos arba rikiavimo taisyklės yra sudėtingesnės nei daugelyje kitų šalių.

EUC 等のエンコーディング方法は日本語と英語が混交しているテキスト

*8 pav*. Japonų rašto pavyzdys. Pirmi trys ženklai lotyniški (jie naudojami rašant tarptautines santrumpas), toliau: nepabraukti ženklai – Hiraganos (ja paprastai rašomi gramatiniai žodžiai ir žodžių galūnės), pabraukti viena linija – Katakanos (ja paprastai rašomi žodžiai atkeliavę iš kitų kalbų) ir pabraukti dviem linijom – Kanji

**Eilučių laužymas**. Taisyklės susiję su eilučių sandara, pavyzdžiui, žodžių skaidymas, teksto kėlimas į sekančią eilutę, skirtingose kultūrose skiriasi. Tam įtakos turi ir raštų skirtumai.

**Skaitmenys**. Taip kaip skiriasi raštų raidės, gali skirtis ir skaitmenys. Vakarų kultūroje daugiausia naudojami arabiški skaitmenys „1, 2, 3, …“, tačiau iš tiesų arabų rašte jie žymi kitokiais ženklais „١, ٢, ٣, …“. Kitų raštų pavyzdžiai: Devanagari „१, २, ३,…“, tailandiečių „๑, ๒, ๓,…“. Pavyzdžiui, kinai turi net du skaitmenų variantus, vienas jų skirtas kasdieniniam naudojimui „一, 二, 三,…“, kitas piniginėms sumoms žymėti „壹, 貳, 參, …“.

[25] Taip vadinamas Japonijoje naudojamas Han raštas.

### 4.2.2. Kalbos

Pasaulyje egzistuoja daugybė skirtingų kalbų (ISO 639-3 apibrėžia 7299 kalbų kodus) [VV77] [Go05]. Kalbų skirtumai jau savaime yra labai svarbūs PĮ internacionalizacijai, nes pagrindinę lokalizavimo darbų dalį sudaro jos tekstų vertimas į kitas kalbas. Tačiau egzistuoja ir mažiau akivaizdūs kultūriniai skirtumai, kurie taip pat turi įtakos kalbos suvokimui. Kaip teigia Sapir [Sa49] ir kiti tyrinėtojai [Wh56] [Vo73] [SY00] ryšys tarp kalbos ir kultūros stiprus ir daugiamatis. Todėl negalime būti tikri, kad netgi tą pačią kalbą naudojantys skirtingų kultūrų žmonės supras išsakytą arba užrašytą informaciją vienareikšmiškai. Pavyzdžiui, Evers [Ev01] tyrimas parodo, kad vienoje kultūroje naudojami tos pačios kalbos išsireiškimai arba metaforos, gali būti klaidingai suprantami kitose kultūroje.

Klaidingai gali būti interpretuojami ne tik kultūriškai priklausomi išsireiškimai arba metaforos, tačiau ir daugeliu požiūrių įprasta informacija. Tam įtakos gali turėti įvairios naudotojų kultūrinės nuostatos. Pavyzdžiui, skirtingose kultūrose, tos pačios prielaidos gali būti siejamos su skirtingais veiksmais (ta pati problema Anglijoje ir Japonijoje gali būti sprendžiama skirtingai – taip kaip priimta tose kultūrose) [Ev01].

PĮ internacionalizavimo metu svarbu užkoduoti informaciją tokiu būdu, kad ji būtų vienodai suprantama ir priimtina bet kurios kultūros žmonėms. Esselink [Es02] teigia, kad didelę įtaką tam turi terminija ir stilius. PĮ lokalizuojantiems žmonėms originalo kalba paprastai nebūna gimtoji, todėl tam, kad jie galėtų lengviau ir be klaidų suprasti verčiamus tekstus šie turi būti parengti laikantis tam tikrų stiliaus ir terminijos reikalavimų. Toliau pateikiamas Esselink suformuotas tokių reikalavimų sąrašas

Terminija:

- Sukurti terminų žodyną į kurį turi būti įtraukti su kuriama PĮ susiję terminai. Šie terminai turi būti nuosekliai naudojami tiek PĮ tekstuose, tiek su pateikiamose dokumentacijose ir žinynuose.
- Tiems patiems veiksmams naudoti tas pačias frazes ir terminiją, tam kad būtų išlaikomas nuoseklumas ir jie tarpusavyje derėtų.

Stilius:

- Tekstas turi būti aiškus ir kiek įmanoma labiau vienareikšmiškas.
- Naudoti trumpus, glaustus ir pageidautina veikiamosios rūšies sakinius.
- Vengti žargono, slengo, humoro, teksto su informacija apie politiką, TV programomis, nacionalinėmis vertybėmis ir pan.
- Vengti informacijos apie gyvūnus, religijas, šventus objektus ar simbolius.
- Vengti informacijos susijusios su metų laikais, laiko juostomis, orais ar šventėmis.
- Vengti santrumpų ir akronimų.
- Atsižvelgti į akivaizdžius kultūrinius skirtumus: matavimo vienetus, datų, laiko formatus, kalendorius ir pan. (pvz. derėtų rašyti datas ilgu formatu).
- Vengti reklamos, kuri kitose šalyse gali būti nepriimtina ar netgi neteisėta.

Tačiau ir šie reikalavimai negali pilnai išspręsti problemų su kuriomis susiduriama verčiant PĮ tekstus. Visų pirma terminai ir frazės dažnai gali būti verčiami į kitas kalbas daugiareikšmiškai. Pavyzdžiui, *From ... to = Kas ... kam* (el. laiške), *Nuo ... iki* (kai kalbama apie intervalą); *check = pažymėti* (varnele), *tikrinti* (pvz., rašybą), *tab = tabuliacijos ženklas* (dokumente), *kortelė* (dialogo lange) ir t. t. [Gr03]. Nevienareikšmiškumų gali padaugėti ir dėl to, kad daugelis tų pačių angliškų žodžių (vienodai rašomų) (remiantis Esselink [Es02] daugiau kaip 80 % lokalizavimo projektų anglų kalba būna pirminė) gali atlikti ir daiktavardžio ir

veiksmažodžio funkcijas, pavyzdžiui, *file = byla, įdėti*; *view = rodinys, rodyti*; *list = sąrašas, išvardyti*; *bookmark = adresas, įrašyti adresą* ir t.t. [Gr03]

Dažnai pasitaiko ir atvejų kai terminai neturi atitikmens kalboje į kurią verčiamas tekstas. Tokia situacija dar daugiau apsunkina lokalizavimą nei terminų daugiareikšmiškumas, nes sukurti geram terminui reikalingi tos srities specialistai ir tai gali užtrukti santykinai ilgai.

Problemų gali sukelti ir žodžių linksnių bei gramatinių formų derinimas. Pavyzdžiui, programose pasitaiko vietų, kur frazės gaunamos sujungiant dvi ar daugiau frazių (žodžių), naudojamų ir kitur: savarankiškai arba kituose frazių junginiuose. Daugelyje programų pasitaikantis pavyzdys, kai jungiamas komandų *Atšaukti* arba *Atstatyti* pavadinimas su pavadinimu tos komandos, kurios veiksmas atšaukiamas arba atstatomas, pavyzdžiui, *Įterpti*, *Iškirpti*. Mechaniškai jungiant gaunamos gramatiškai nesuderinamos žodžių poros: *Atšaukti Įterpti*, *Atstatyti Iškirpti*. Pagal lietuvių kalbos taisykles taisyklingas vertimas būtų: *Atšaukti įterpimą*, *Atstatyti iškirpimą*. Panašus pavyzdys su daiktavardžių linksnių derinimu prie kintamų skaičių (kintamųjų reikšmių), pavyzdžiui, 1 daiktas, 2 daiktai, 10 daiktų. Dažniausiai yra galimybė pateikti tik viena daiktavardžio formą, geriausiu atveju – dvi, nes tiek formų yra anglų kalboje, o lietuvių kalboje reikia trijų.

Kaip rodo praktika didžiausias vertimo problemas kaip tik ir sukelia neaiškių (daugiareikšmių arba turinčių kultūrinį atspalvį) terminų arba sakinių vertimas. Teksto vertimą galima išskirti į du etapus: juodraštinį vertimą ir vertimo derinimą. Juodraštinio vertimo metu tekstas tiesiog išverčiamas. Tačiau didesnę darbų dalį sudaro vėlesnis šio vertimo derinimas. Derinant sukuriamos situacijos, kad visos galimos frazės pasirodytų ekrane (nors tai ne visada pavyksta). Jo metu išryškėja vertimo netikslumai, kurių daugumos nebuvo galima numatyti pirmajame etape. Nes pavyzdžiui, verčiant tekstus iš anglų kalbos į lietuvių pirmojo etapo metu dažnai nebūna aišku kurią termino reikšmę, linksnį ar gramatinę formą pasirinkti. Tai tampa aišku tik pamačius tekstą realiame kontekste.

## 4.3. Lokalės

Kultūrinių nuostatų registravimo procedūrų standarte (LST ISO/IEC 15897) pateikiamas toks lokalės apibrėžimas: **lokalė** yra *„naudotojo aplinkos poaibio, priklausančio nuo kalbos ir kultūros normų, apibrėžimas“*.

Kalbant apie programų internacionalizavimą arba lokalizavimą, lokalės terminas dažnai naudojamas ir šiek tiek platesne prasme, kaip *„visų nuo konkrečioje vietovėje (paprastai tai yra šalis arba provincija) priklausančių dialogo su vartotoju kultūrinių ir kalbinių aspektų visuma*“. Deja, formaliai apibrėžti visus šiuos aspektus neįmanoma. Pavyzdžiui, neįmanoma apibrėžti paveikslų kultūrinių aspektų, galima tik pateikti rekomendacijas kaip parengti paveikslus, kad jie būtų priimtini daugumoje kultūrų. Todėl aprašant lokales formaliu būdu paprastai apsiribojama apibendrinta kultūrinių aspektų aibe (ji iš esmės atitinka pirmajame skyriuje nagrinėtų akivaizdžių kultūrinių faktorių aibę), o pagrindinės dalys, kurias apima dauguma standartinių lokalių, yra:

- Kalba ir šalis. Kokia kalba naudotojas palaiko dialogą su programine įranga? Kokie yra kalbos ir šalies pavadinimai, bei angliški jų vertimai?
- Ženklų kodavimas, klasifikacija ir rikiavimas. Kurie ženklai yra raidės, skaičiai arba skyrybos ženklai? Ar kalba turi didžiąsias ir mažąsias raides ir kaip, jos tarpusavyje konvertuojamos? Kokios yra naudojamos ženklų koduotės ir kada jos taikomos?
- Formatai (skaičių, datų ir laiko, valiutos, adresų, telefono numerių). Kaip yra vaizduojami sveikieji ir realieji skaičiai bei piniginės sumos, datos ir laikas? Kokia tvarka rašomi adresai ir telefonų numeriai?

- Kalendorius, laiko juosta. Koks kalendorius yra naudojamas? Ar jame yra skaičiuojamos dienos arba savaitės dienos ir kaip? Informacija apie vasaros ir žiemos laiką.
- Matavimo vienetai, popieriaus lapo formatai. Kokie matavimo vienetai naudojami matuojant ilgį, svorį, garsą, greitį ir kt.?

Skirtingi PĮ gamintojai naudojo skirtingus metodus kultūrinėms nuostatoms aprašyti, todėl atsirado keletas standartinių lokalių modelių, kurie nėra tarpusavyje pilnai suderinami. Vieni jų apibrėžti tarptautiniais standartais: POSIX (IEEE Standard 1003, ISO/IEC 9945), C++ (ISO/IEC 14882), FDCC-set (ISO/IEC 14652), kiti laikomi standartais de-facto: *Windows*, Java lokalių modeliai [La03].

Pagrindiniai lokalę identifikuojantys elementai yra kalba ir šalis. Pavyzdžiui, JAV ir Jungtinėje Karalystėje vartojama ta pati kalba, tačiau jos turi skirtingas lokales, nes skiriasi jų kultūrinės nuostatos. Atvirkščiai yra Suomijoje, joje vartojamos dvi valstybinės kalbos (suomių ir švedų), tad reikia ir dviejų skirtingų lokalių. Lokalių identifikavimui kompiuteryje paprastai naudojama standartais ISO 639 ir ISO 3166 apibrėžtų kalbos ir šalies kodų kombinacija. Tokią lokalių identifikaciją apibrėžia RFC 3066 ir XPG4 dokumentai, ir ji yra plačiai taikoma. Nors, pavyzdžiui, *Windows* aplinkoje lokalės identifikuojamos skaitiniais lokalių identifikatoriais LCID[26] (pavyzdžiui, pagal XPG4 lietuvių lokalė identifikuojama eilute „lt-LT“, o pagal LCID skaičiumi $427_{16}$) [TDD95].

Lokalių palaikymą kuriant PĮ paprasčiausia realizuoti pasinaudojimas platformos lokalių paslaugomis. Tačiau šis būdas nėra universalus, dėl dviejų pagrindinių priežasčių:

1. Platformos lokalėse paprastai apibrėžiama gana siaura kultūrinių nuostatų aibė kurios kuriant platesnių galimybių PĮ gali nepakakti. Pavyzdžiui, gana daug papildomų lokalių elementų gali būti naudojama biuro programose – įvairūs datų, adresų ir kt. formatai, dokumentų šablonai ir pan. [LS01] [LD03].
2. Skirtingų platformų lokalių ir jų API realizacija paprastai būna nesuderinama. Pavyzdžiui, *Unix* OS naudoja POSIX lokalių modelį ir jam pritaikytą API, kurie iš esmės skiriasi nuo naudojamų *Windows* OS.

Dėl šių priežasčių, kuriant PĮ kartais yra paprasčiau naudoti savus lokalės duomenis. Tam galima panaudoti jau sukauptus, laisvai platinamus, tarptautinėse duomenų bazėse esančius lokalių duomenis. Pagrindinės šiuo metu egzistuojančios lokalių duomenų bazės yra dvi. Lokalių duomenys jose yra kaupiami remiantis tarptautiniais standartais:

1. Kultūros elementų registravimo procedūrų standartu ISO/IEC 15897.

   Šis standartas apibrėžia kultūros elementų registravimo tvarką, aprašant ne tik formaliu būdu (remiantis POSIX lokale arba kompiuterio – nagrinėjamu formatu, pavyzdžiui XML), tačiau ir neformaliu – nupasakojant. Lokalių duomenys yra kaupiami tinklalapyje http://std.dkuug.dk/cultreg/.

2. Unikodo techniniu standartu UTS 35 LDML (Locale Data Markup Language).

   Šis standartas pateikia formalizuotą lokalės duomenų aprašymo metodą, paremtą XML. Tai gana naujas standartas, sukurtas 2004 m. Remiantis šiuo standartu lokalių duomenys yra kaupiami CDLR (Common Locale Data Repository) duomenų bazėje (tinkalapis http://unicode.org/cldr/). Duomenys šioje duomenų bazėje yra kaupiami sėkmingiau, lyginant su pirmąja (joje duomenys kaupiami savanoriškai juos pateikiant), nes jį iniciavo ir duomenų kaupimu užsiima tuo suinteresuoti PĮ gamintojai, kurie iš anksto yra numatę aiškų ir išsamų veiksmų planą, kaip tai atlikti.

[26] LCID – locale identifiers. Jų reikšmės pateiktos http://www.microsoft.com/globaldev/reference/lcid-all.mspx

Deja aprašyti visus naudotojo aplinkos poaibio elementus, priklausančius nuo kalbos ir kultūros normų neįmanoma. Kaip jau minėta egzistuoja daugybė neformalių kalbinių arba kultūrinių faktorių. Todėl neįmanoma sukurti PĮ internacionalizavimo metodo, kuris leistų PĮ automatiškai prisitaikyti prie visų konkrečioje kalbinėje arba kultūrinėje terpėje galiojančių nuostatų. Todėl lokalizuotojų įsikišimo galutiniame PĮ pritaikymo etape poreikio neįmanoma panaikinti.

## 4.4. Duomenų kodavimas

Dažnai pasitaikanti klaida kuriant PĮ yra netinkamo teksto kodavimo metodo pasirinkimas. Iš dalies tai galima paaiškinti istorinėmis kompiuterių raidos priežastimis. Dėl kompiuterių sandaros savybių duomenų kodavimas 8 bitais leido jų galimybes išnaudoti efektyviausiai. Todėl gana ilgai jis buvo naudojamas ir teksto kodavimui. Tačiau šiuo metu tiek kompiuteriai, tiek platformos jau yra pakankamai patobulėjusios ir nebesudaro kliūčių naudoti šiuolaikinius, universalius kodavimo metodus.

Pagrindinis standartas apibrėžianti 8 bitų kodų lenteles yra ISO 8859. Šiuo metu jį sudaro 16 dalių apimančių įvairius regionus ir atsiradus poreikiui jis gali būti papildytas naujomis dalimis. 8 bitų kodavimo lenteles taip pat turi dalis šio standarto nesilaikančių OS, pavyzdžiui – DOS, *Windows*, *Mac OS*.

8 bitais galima užkoduoti 255 ženklus ir to pakanka daugumos raštų ženklams užkoduoti naudojant skirtingas kodavimo lenteles. Tačiau to nepakanka raštams kurie naudoja daugiau rašto ženklų (pvz. logo-skiemeniniams raštams). Be to teksto kodavimas 8 bitais sukelia problemų perkodavimo iš vienos kodų lentelės į kitą metu, nes tokiu būdu gali būti prarandama dalis informacijos. Todėl internacionalizuota PĮ privalo naudoti universalų duomenų kodavimo metodą, neturintį tokių trūkumų.

Universalų duomenų kodavimo metodą pateikia Unikodo standartas. Jį 1991 metais sukūrė ir toliau tobulina to paties vardo konsorciumas. Jis apibrėžia pakankamai didelę ženklų kodų lentelę (apima 1 114 112 kodo vienetų) į kurią telpa visi pasaulyje naudojami rašto ženklai.

Unikodo standarte apibrėžta kodų lentelė yra ekvivalenti ISO 10646 standarto poaibiui UCS-2. Skirtingai nei ISO 10646 standartas Unikodas ne vien tik aprašo ženklų aibę, bet ir pateikia taisykles bei specifikacijas susijusias su praktiniu standarto taikymu – pavyzdžiui, kombinacinių sekų normalizavimo, dvikrypčio teksto vaizdavimo specifikacijas ir pan. Dėl šios priežasties Unikodo standartas tapo plačiau taikomas ir žinomas. Jį palaiko ir dauguma šiuolaikinių platformų.

Unikodo standartas pateikia tris ženklų kodavimo metodus UTF-32, UTF-16 ir UTF-8. Skaičius metodo pavadinime reiškia bitų skiriamų išreikšti kodo vienetui skaičių. Tik 32 bitų pakanka, kad visus Unikodo ženklus būtų galima išreikšti atskirais kodo vienetais. Tuo tarpu naudojant kitus metodus dalis ženklų išreiškiama kodo vienetų sekomis. Pavyzdžiui, 16 bitų atveju atskirais kodo vienetais koduojami pagrindiniame lakšte esantys ženklai, o likę ženklai išreiškiami surogatų srityje esančių kodo vienetų poromis. Ženklai esantys už pagrindinio lakšto ribų naudojami labai retai, todėl šis metodas yra gana efektyvus ir labai plačiai naudojamas. 8 bitų atveju pusė kodo vienetų skiriama užkoduoti ASCII ženklams, o likę ženklai koduojami 2-6 kodo vienetų sekomis. Lyginant tarpusavyje Unikodo kodavimo metodus realizacijos atžvilgiu juose galima įžvelgti tiek trūkumų, tiek pranašumų, tačiau internacionalizacijos požiūriu jie yra lygiaverčiai.

Unikodo standartas yra pranašesnis už kitus ženklų kodavimo standartus ne tik dėl universalumo, tačiau ir dėl daugelio kitų gerų savybių: efektyvumo (Unikodu koduotą tekstą lengva analizuoti ir apdoroti), aiškios ženklų semantikos, vienareikšmiškumo (įvairiose raštuose

esantys tokie patys ženklai nepasikartoja), konvertabilumo (užtikrinamas tikslus ženklų konvertavimas tarp Unikodo ir kitų plačiai naudojamų standartų) ir kt. [Un03].

Be paties standarto Unikodo realizacijai kompiuteryje, taip pat labai svarbūs jo priedai, bei techniniai standartai.

## 4.5. Išteklių atskyrimas

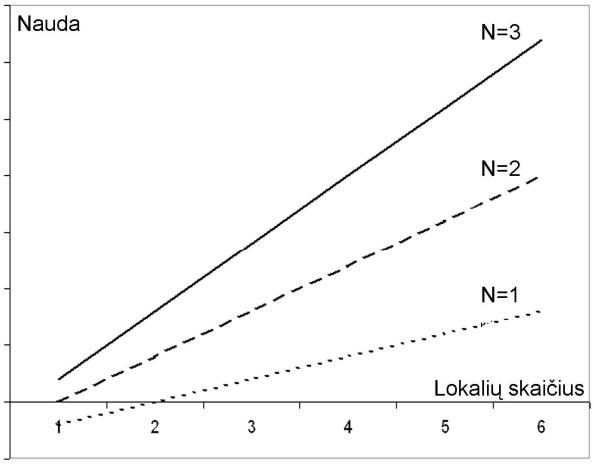
Pagal išteklių atskyrimo būdą Taylor [Ta93] išskiria tris internacionalizacijos tipus: 1) kompiliavimo meto internacionalizacija; 2) susaistymo meto internacionalizacija 3) vykdymo meto internacionalizacija.

**Kompiliavimo meto internacionalizacija**[27]**.** Lokalizacijos atsiradimo pradžioje, kol dar nebuvo sukurta šiuolaikinių internacionalizavimo metodų, PĮ dažniausiais buvo lokalizuojama pirminio teksto lygyje, t. y. padarant atskiras jos pirminio teksto kopijas kiekvienai lokalei ir jas lokalizuojant. Žinoma toks būdas yra labai neefektyvus, nes lokalizuojamų išteklių pakeitimai reikalauja pilnai perkompiliuoti PĮ, taip pat testuoti ir taisyti tenka kelias kopijas vienu metu. Šis būdas taip pat nepatikimas klaidų atžvilgiu, nes atsiranda tikimybė, kad pirminis tekstas gali būti pažeistas lokalizavimo metu.

Nors šis būdas laikomas pasenusiu, tačiau jį vis dar dažnai naudoja atvirosios programos [LD03]. Šiuo atveju galima išskirti dvi galimas lokalizavimo strategijas:

1. Lokalizuoti tiesiogiai, nekeičiant esamos internacionalizacijos.
2. Perprogramuoti internacionalizaciją į susaistymo arba vykdymo meto internacionalizacija.

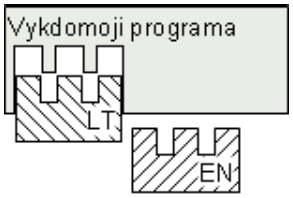
Pagrindinė priežastis dėl ko gali būti pasirinkta pirmoji strategija yra ta, kad pirmoji lokalizuoto produkto versija gaunama žymiai sparčiau. Antroji strategija pradžioje reikalauja daugiau investicijų, tačiau tolesnis lokalizavimas yra žymiai paprastesnis. Todėl šios investicijos lokalizuojant sekančias versijas atsiperka ir duoda naudos [HH97]. 11 paveiksle pavaizduota naudos priklausomybė nuo palaikomų lokalių skaičiaus ir išleidžiamų PĮ versijų.




*9 pav*. Internacionalizavimo nauda

[27], Šis tipas turi mažai ką bendro su šiuolaikine internacionalizacijos samprata, tačiau pavadintas internacionalizacija derinantis prie Taylor terminijos.

**Susaistymo arba vykdymo meto internacionalizacija**. Nuo to laiko kai buvo parašyta Taylor [Ta93] knyga susaistymo ir vykdymo meto internacionalizacijų būdai suartėjo, todėl čia jie nagrinėjami kartu. Šiuo atveju ištekliai lokalizuojami ne pirminio PĮ teksto lygyje, bet jau sukompiliuoto dvejetainio kodo lygyje. Jie gali būti įkompiliuoti į vykdomąsias programas arba į vykdymo metu prie jų jungiamas išteklių bibliotekas. Pažymėtina, kad antrasis variantas yra pranašesnis, nes tokiu būdu galima kartu pateikti daugiau nei vienos lokalės išteklių bibliotekas ir taip sudaryti vartotojui geresnes galimybes pasirinkti norimą lokalę.




*10 pav*. Vykdomoji programa ir su ja susiejamos išteklių bibliotekos

Lokalizuojant šiuo būdu internacionalizuotą PĮ išteklių pakeitimai nereikalauja jos pirminio teksto perkompiliavimo, pakanka perkompiliuoti tik pakeistus išteklius. Todėl tokios PĮ lokalizavimas yra žymiai paprastesnis ir nekyla pavojaus lokalizavimo metu ją pažeisti.

Šiuo būdu atskirtus išteklių failus galima saugiai lokalizuoti, nesibaiminant pažeisti PĮ, nes pagrindinis jo principas atskirti išteklius taip, kad jie neturėtų įtakos vykdomųjų programų funkcionalumui.

Dar vienas labai svarbus šio metodo pranašumas: atskirtus išteklių failus galima patikėti lokalizuoti tame besispecializuojančioms bendrovėms, kurios gali tai atlikti kokybiškiau, greičiau ir pigiau, nei tai būtų lokalizuojant pačiam gamintojui. Lokalizavimas mašininio kodo lygyje užtikrina, kad nebus pažeistos PĮ pirminio teksto autorinės teisės.

Pagrindiniai šiuo metu naudojami išteklių atskyrimo metodai yra:

1. GetText.
2. *.RC*.
3. *.RESX*.
4. ResourceBundles.

**GetText** metodas remiasi lokalizuojamų išteklių atskyrimu į išteklių katalogus naudojant GNU GetText priemonių rinkinį [DMP02] [DVW03]. Iš esmės šios priemonės skirtos atskirtis nuo pirminio teksto tik sąsajos tekstą ir pranešimus į teksto katalogus saugomus tam skirtuose .mo formato failuose. Tuo tarpu kitų išteklių (dialogų šablonų, paveikslų ir kt.) atskyrimui specialių metodų jis nenumato ir tai yra pagrindinis jo trūkumas. Kadangi metodas daugiausia taikomas kuriant atvirąsias programas (daugiausia skirtas Unix sistemai), tai šių elementų lokalizavimas dažniausiai atliekamas pirminio teksto lygyje, kas kaip jau minėta nėra efektyvu ir patikima.

Sudarant teksto katalogus GetText metodas leidžia naudoti nevienareikšmius, tiesioginius lokalizuojamo teksto identifikatorius. T.y. programuotojui leidžiama nurodyti lokalizuojamą tekstą jį apgaubiant funkcija „_“ (pvz. Pascal kalboje _(‘Lokalizuojamas tekstas’)). Tokiu atveju identifikatoriumi tampa tas pats tekstas. Minėta funkcija turi dvejopą paskirtį: 1) programos vykdymo metu ji pakeičia tekstą lokalizuotu; 2) taip pažymimas tekstas, kuris turi būti išskirtas iš pirminio teksto (tai gali būti atliekama naudojant specialiai tam sukurtus pirminio teksto analizatorius). Galimybė naudoti tiesioginius lokalizuotino teksto identifikatorius ne tik palengvina programuotojo užduotį, bet gali sukelti ir problemų. Dažnai pasitaiko atvejų, kad

tekstas yra verčiamas nevienareikšmiškai (pavyzdžiui, anglų kalbos meniu komanda „View“ gali būti verčiama kaip daiktavardis „Rodinys“ ir kaip veiksmažodis „Rodyti“), o GetText metodas tokios galimybės nenumato – programuotojas asmeniškai turi ieškoti būdų kaip tokias problemas spręsti.

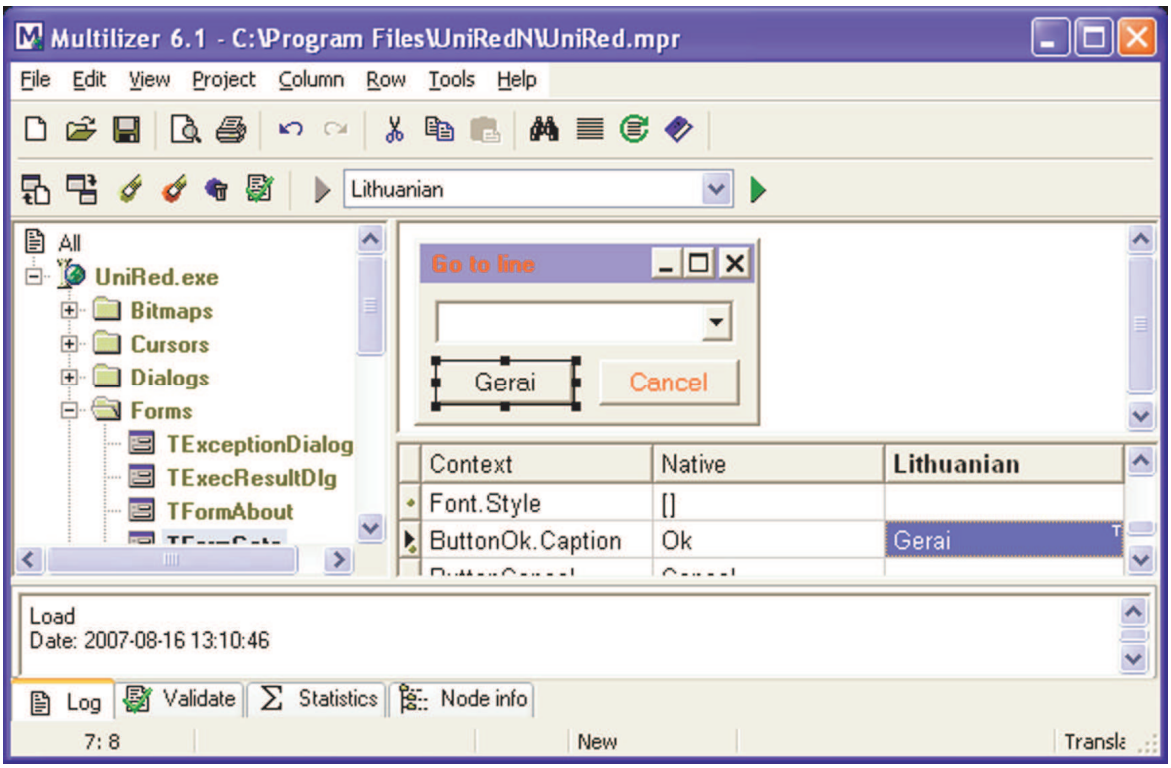
Gera GetText metodo savybė, kurios neturi kiti metodai, yra ta, kad jis numato galimybę kaityti žodžių linksnius atitinkamai pagal į tekstą įterpiamus skaičius. Tam į išteklių failus įterpti scenarijus, kurių pagalba galima realizuoti įvairiose kalbose naudojamas linksnių kaitymo taisykles.

***.RC*** metodas remiasi išteklių įkompiliavimu į vykdomųjų failų arba dinaminių bibliotekų išteklių sekcijas. Tokiam išteklių įkompiliavimui seniau programuotojai turėdavo juos aprašyti specialaus .RC formato faile, iš to ir kilo metodo pavadinimas. Tuo tarpu šiuolaikinės vizualaus programavimo priemonės daugumą veiksmų atlieka automatiškai, o lokalizuojami ištekliai gali būti pažymimi naudojant tam skirtas programavimo kalbų konstrukcijas.

Skirtingai nuo GetText šis metodas numato ne tik teksto, bet ir visų kitų lokalizuojamų išteklių atskyrimo galimybę. Tuo šis metodas yra pranašesnis už kitus. Tai sudaro galimybes:

1. Lokalizuoti visus sąsajos elementus.
2. Taikyti vizualaus lokalizavimo priemones.

Vizualaus lokalizavimo priemonės leidžia lokalizavimo metu matyti kaip atrodys jau lokalizuota PĮ sąsaja. Tai suteikia galimybes sparčiau ir tiksliau atlikti sąsajos teksto vertimą, nes yra iš karto matomas teksto frazių kontekstas ir todėl mažiau laiko sugaištama jų derinimui (pav. 13).




*11 pav.* Vizualaus lokalizavimo priemonės pavyzdys. Kairėje lango pusėje matyti išteklių medis, kurio šakos atitinka išteklių sekcijas ir jų elementus. Pasirinkti elementai matomi dešinėje lango pusėje kur tuo pačiu juos galima ir taisyti

Be 15 standartinių išteklių tipų (žymekliai, paveikslėliai, piktogramos, meniu, dialogai ir kt.) metodas numato ir nestandartinių išteklių atskyrimo galimybę. Tam yra skiriama RCDATA išteklių sekcija [Mi03]. Pavyzdžiui, šią galimybę naudoja Delphi programavimo sistema dialogų šablonų įkompiliavimui specialiu, jos naudojamu DFM formatu. Nors šis formatas nėra standartinis, tačiau dauguma vizualaus lokalizavimo priemonių, buvo jam pritaikytos ir geba taip koduotus dialogų šablonus teisingai parodyti.

Nors paprastai šis metodas siejamas su *Windows* OS skirta PĮ, tačiau jis gali būti taikomas ir kitų OS. Pavyzdžiui, *Unix* (*Linux*) OS vykdomųjų failų ir bibliotekų formatas labai mažai skiriasi, nuo naudojamų *Windows* OS ir šis metodas jose taip pat gali būti sėkmingai taikomas [Th01] [TIS95] [Mi99].

***.RESX*** metodas skiriasi nuo *.RC* tik tuo, kad ištekliai yra aprašomi ne *.RC* formato, o *.RESX* formato paremto XML technologija failuose. Tai žymiai pažangesnė technologija leidžianti tame pačiame faile aprašyti įvairių tipų išteklius (dvejetainiams ištekliams, pavyzdžiui paveikslams, koduoti yra naudojamas Base64 metodas (RFC 2045)) [Mi06]. Šis išteklių aprašymo būdas panašus į būdą naudojamą apsikeitimo vertimų atmintimi standarte XLIFF [OASIS03]). Nors metodas šiuo metu naudojamas išskirtinai tik kuriant *.NET* platformoms skirtą PĮ, tačiau idėja yra gana pažangi ir dėl XML technologijos atvirumo gali būti taikoma ir kitais atvejais.

**ResourceBundles** metodas ištekliams atskirti naudoja specialaus formato tekstinius failus. Juose ištekliai gali būti aprašomi paprasčiausiai priskiriant jiem identifikatorius arba *Java* klasių pagalba. Tokiu būdu galima aprašyti ne tik statiškus išteklius, kaip tekstas ar paveikslai, tačiau ir dinaminius, realizuojant juos kaip klasės metodus. Ištekliai yra įkeliami naudojant *Java* API. Tam skirta `ResourceBundle` objektų klasė [Co98].

Pagrindinis šio metodo trūkumas, kad ištekliai aprašomi gana neefektyviu būdu. Net ir atskiriamo teksto eilučių aprašymui turi būti sukuriami atskiri failai, eilutėms priskiriami identifikatoriai, pirminiame programų tekste specialių metodų pagalba aprašomas kiekvienos eilutės įkėlimas. Tai darbas reikalaujantis didelio programą kuriančio programuotojo indėlio, nors jis gali būti automatizuotai atliekamas paties kompiliatoriaus.

Šis metodas yra gana universalus ir gali būti naudojamas ne tik programuojant Java. Panašiai išteklių atskyrimas realizuojamas ICU projekte, nors jo komponentai gali būti naudojami ir programuojant C, bei C++ kalbomis [IBM05].

## 4.6. Leksikos elementai

Nors oficialiuose programavimo kalbų aprašuose paprastai nekalbama apie leksikos elementų lokalizavimo galimybes, tačiau dažnai numatoma galimybė, kad jie gali būti parinkti realizacijos metu. Tokiu atveju šiuos žymenis galima parinkti taip, kad jie atitiktų konkrečios lokalės kalbinę ir kultūrinę terpę. Praktiškai tai būtų kalbos lokalizavimas ir kuriant kompiliatorių reikėtų pasirūpinti, kad leksikos elementus būtų lengva pakeisti, t.y. jį internacionalizuoti [Gr00].

Programavimo kalbų lokalizavimas dar nėra įprastas reiškinys, egzistuoja labai mažai internacionalizuotų kompiliatorių, kurie leistų jas lokalizuoti. Viena iš žinomiausių Lietuvoje ir pasaulyje naudojamų programavimo kalbų, kurių bazinius žodžius įprasta lokalizuoti yra LOGO. Lietuvoje ji naudojama bendrojo lavinimo mokyklose, mokant algoritmavimo žemesnių klasių mokinius. Lokalizavimas suteikia jai pranašumų prieš nelokalizuotas kalbas, nes lokalizuotos kalbos leksika yra aiškesnė, paprastesnė, lengviau įsimenama ir išmokstama. Lokalizuotos kalbos elementai užrašomi gimtąja kalba yra natūralesni ir lengviau suvokiami. Be to, naudoti mokymui sulietuvintą PĮ reikalauja ir valstybinės kalbos įstatymas, kurio 11 straipsnyje

teigiama: „Valstybė garantuoja Lietuvos Respublikos gyventojams teisę įgyti bendrąjį, profesinį, aukštesnįjį ir aukštąjį išsilavinimą valstybine kalba“.

Panašių tikslų siekiama ir lokalizuojant kitą Lietuvos bendrojo lavinimo mokyklose algoritmavimui mokyti naudojama sistemą – *Free Pascal*. Ją lokalizuojant, susidurta su problemomis, nes *Free Pascal* kompiliatorius nėra pakankamai internacionalizuotas. Kadangi tai atvirojo teksto kompiliatorius, tai nuspręsta jį internacionalizuoti patiems ir buvo atliktas tyrimas, kaip tai atlikti. Tyrimas atskleidė daug su kompiliatorių internacionalizacija susijusios, naudingos informacijos [LD01] [La05].

Egzistuoja svarbus ryšys tarp žmogaus mastymo ir kalbos. Žmogus naudoja kalbą, kad mintyse apibrėžtai išreikštų tai, ką galvoja. Taip pat ir programuotojas, kurdamas programą mąsto programavimo kalbos, kuria programuoja, konstrukcijomis ir sąvokomis. Todėl labai svarbu, kad šios sąvokos (taip pat žyminčios įvairius leksikos elementus) būtų kuo artimesnės gimtajai kalbai.

Kalbų lokalizavimas ypatingai svarbus tautoms, kurios naudoja ne lotynišką rašto ženklų sistemą (kirilicą, arabišką arba kt.). Nes tuomet kyla nepatogumų norint surinkti lotyniškais rašmenimis pažymėtus leksikos elementus. Taip pat, teksto rinkimo problemų gali iškilti ir lotynišką rašto ženklų sistemą naudojančioms tautoms, kurios nenaudoja visų anglų abėcėlės raidžių – pavyzdžiui, lietuvių abėcėlėje nėra „w“, „q“, „x“ raidžių.

Viena priežasčių, kodėl vengiama lokalizuoti programavimo kalbas yra galimos suderinamumo problemos tarp lokalizuotų kompiliatorių. Tačiau šią problemą galima spręsti lokalizuojamus elementus aprašant formaliai. Pavyzdžiui, tai galima atlikti įtraukus juos į lokalių aprašus. Taip pat reikėtų vertimus standartizuoti. Tai užkirstų kelią galimoms skirtingoms programavimo kalbų interpretacijoms jas lokalizuojant.

Į lokalių aprašus įtrauktini šie leksikos elementai: baziniai žodžiai, operacijų ženklai, skaičiai, skyrybos ženklai, standartiniai vardai. Tam, kad juos būtų galima pilnavertiškai lokalizuoti kompiliatorius turi palaikyti Unikodo standartą. Unikodo standartas apibrėžia ne tik universalų ženklų kodavimo metodą, tačiau pateikia ir programavimo kalbose naudojamų vardų sudarymo sintaksės taisykles. Juo remiantis, vardai gali būti sudaromi tiek iš lotyniškos abėcėlės, tiek kitų abėcėlių (kirilicos, graikų, arabų, kinų ir kt.) raidžių bei kitokių ženklų (UAX #31).

UAX #31 bazinę vardų sintaksę apibrėžia tokia taisykle:

```
<vardas> ::= <pirmas vardo simbolis>(<tolesnis vardo simbolis>)*
```

Joje simbolis <pirmas vardo simbolis> reiškia bet kokią raidę, o <tolesnis vardo simbolis> – bet kokią raidę bei dešimtainius skaitmenis, netarpo ženlus, kombinacinius ženklus, jungiamuosius skirybos ženklus. Standartas griežtai neriboja vardų sudarymo sintaksės ir leidžia be minėtų ženklų naudoti programavimo kalbose jau priimtus kitokio tipo ženklus. Pavyzdžiui, $, @, #, arba _ (pastarąjį ženklą įprasta naudoti daugelyje programavimo kalbų).

Standartas taip pat nusako alternatyvių vardų sudarymo taisykles. Iš esmės jų sudarymui naudojamų ženklų gali būti visai neribojama, t.y. vardai gali būti sudaromi iš bet kokių netgi „nasamoningų“ ženklų, remiantis tuo, kad žmogus sudaro vardus supratingai ir „nesąmoningų“ ženklų nenaudos. Tuo tarpu mažesni sintaksės ribojamai leistų sukurti paprastesnį ir efektyvesnį jos analizatorių.

**Baziniai žodžiai**. Daugelyje programavimo kalbų baziniai žodžiai yra anglų kalbos žodžiai arba jų santrumpos. Egzistuoja nuomonė, kad jie turėtų atlikti unifikavimo vaidmenį tarp skirtingų kalbų ir galbūt nederėtų jų lokalizuoti. Tačiau, Grigo [Gr00] atliktas tyrimas parodo, jog iš tiesų bazinių žodžių įvairovė tarp skirtingų kalbų yra labai didelė (tik 3,3 % bazinių žodžių

naudota visose 13 nagrinėtų panašios paskirties kalbų, o 56 % sutinkami tik atskirose kalbose) ir todėl ši nuomonė neturi pagrindo.

Bazinių žodžių internacionalizavimas nėra sudėtingas, nes jie programavimo kalboje teatlieka simbolių žyminčių jos sintaksę vaidmenį. Todėl, tereikia kompiliatoriuje įdiegti mechanizmą, kuris dinamiškai įkeltų lokalizuotus bazinius žodžius iš nustatytos lokalės.

**Operacijų ženklai**. Dalis operacijų ženklų programavimo kalbose yra perimta iš matematikos ir yra tarptautiniai. Tačiau ir jie turi išimčių, pavyzdžiui, kai kurios tautos naudoja skirtingus dalybos arba daugybos ženklus.

Dauguma programavimo kalbų buvo kuriama remiantis ASCII kodų lentele, kurioje yra tik keli, pagrindiniai operacijų ženklai, todėl dalį ženklų teko užkoduoti ženklų poromis (`:=`, `<>`, ir pan.) arba pažymėti vardais (`div`, `or`, ir pan.). Internacionalizavus kompiliatorių, atsiras galimybė pereiti prie natūralaus operacijų žymėjimo matematiniais ženklais (1 lentelė). Tačiau, siekiant nesukelti teksto rinkimo problemų, tikslinga matematinius ženklus įdiegti sinonimiškai seniesiems, o vardais pažymėtas operacijas, lokalizuoti.

*8 lentelė*. Operacijų ženklų pavyzdžiai

| **Operacija** | **Pascal kalboje naudojami ženklai** | **Matematinių ženklų pavyzdžiai** |
|---|---|---|
| Priskyrimas | `:=` | ← (U+2190) |
| Sveikųjų sk. dalyba | `div` | ÷ (U+00f7), \ (U+0092) |
| Daugyba | `*` | · (U+2219), × (U+00d7) |

**Skaičiai.** Skaitmenys, kaip ir rašto ženklai, skirtingose tautose gali skirtis (2 lentelė). Todėl jų lokalizavimo galimybės taip pat turėtų būti numatytos.

*9 lentelė*. Skaitmenų pavyzdžiai

| **Lotynų** | **Arabų** | **Tailando** |
|---|---|---|
| 5 | ٥(U+0665) | ๕ (U+0E55) |
| 7 | ٧(U+0667) | ๗ (U+0E57) |
| 9 | ٩(U+0669) | ๙ (U+0E59) |

Problemų kelia ir skaičių rašymo formatai. Pavyzdžiui, dešimtainės trupmenos skyriklis. Pavyzdžiui, JAV jo vaidmenį atlieka taškas, o daugumoje Europos tautų – kablelis. Paprastai programavimo kalbose kablelis, atlieka ir sąrašų elementų skyriklio funkciją. Todėl, panaudojus kablelį kaip dešimtainės trupmenos skyriklį, gautume dviprasmiškumą. Pavyzdžiui, užrašas 0,12 galės būti traktuojamas ir kaip dešimtainė trupmena ir kaip dviejų skaičių sąrašas. Tačiau, užrašas 0, 12 yra vienareikšmis, nes po kablelio yra tarpas, ir tokia seka neleidžiama rašant skaičius. Tad šią problemą galima išspręsti praplečiant programavimo kalbos sintaksę, pagal kurią, pirmasis užrašas galėtų būti traktuojamas tik kaip skaičius [Gr00].

Panašių problemų sukelia ir skaitmenų grupavimas. Įvairiose kultūrose skaitmenys grupuojami įvairiai. Grupių skyriklio vaidmenį taip pat gali atlikti įvairūs ženklai. Paprastai programavimo kalbos nenumato jokių skaitmenų grupavimo galimybių, nors šiek tiek praplečiant jų sintaksę, skaitmenų grupavimą įdiegti nebūtu sunku.

**Skyrybos ženklai**. Be jau minėtų skaičių dešimtainės trupmenos skyriklio ir grupavimo ženklų, egzistuoja ir kitų skyrybos ženklų grupės (pvz., tarpas gali būti paprastas (U+0020), jungiamasis (U+00A0), ideografinis (U+3000) ir kt.). Tokių ženklų nėra daug, internacionalizuotas kompiliatorius privalo juos suprasti teisingai.

Kiti skyrybos ženklas programavimo kalbose atlieka leksikos elementų vaidmenį. Pavyzdžiui, skliaustai – elementų grupavimo, komentarų žymėjimo ir pan., kabutės – žymi eilučių konstantas. Reikia pastebėti, kad įvairiose lokalėse tam taip pat naudojami įvairūs ženklai. Pavyzdžiui, kabutės gali turėti įvairius užraitymus, būti rašomos viršuje arba apačioje ir pan. (“angliškos”, „lietuviškos“, 「kiniškos」, ″kiniškos2„ ) [Un03].

*10 lentelė*. Skaitmenų grupavimo pavyzdžiai

| **Tautos** | **Pavyzdys** | **Pastabos** |
|---|---|---|
| Anglai | 1,234,567,890.12 | Grupuojami po 3 skaitmenis. Grupės išskiriamos įvairiais ženklais, priklausomai nuo lokalės |
| Vokiečiai | 1.234.567.890,12 | |
| Šveicarai | 1'234'567'890,12 | |
| Lietuviai | 1 234 567 890,12 | |
| Kinai, Japonai | 12億3456万7890.12 arba 12億3,456万7,890.12 | Grupuojami po 4 skaitmenis, išskiriant ideografiniais ženklais. Kartais gali būti išskiriami ir tūkstančiai |
| Indai | 1,23,45,67,890.12 | Grupuojami po 3 skaitmenis pirmoje grupėje, ir po 2 sekančiose, išskiriant kableliais |

**Standartiniai vardai**. Standartinius duomenų tipus, konstantas, modifikatorius galima aprašyti iš naujo. Todėl, internacionalizuojant kompiliatorių, galima įdiegti gana nesudėtingą mechanizmą, kuris šiuos duomenų tipus ir konstantas aprašytų iš naujo, lokalizuotais vardais.

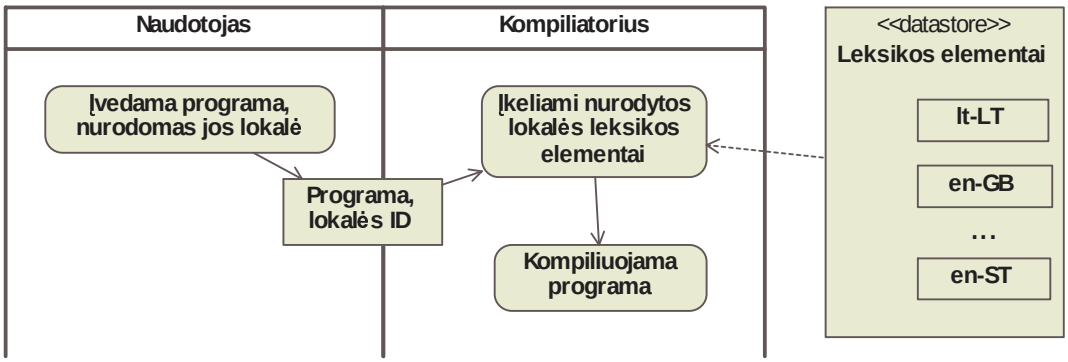
Lokalizuoti standartinių funkcijų vardus šiek tiek sudėtingiau. Jos įvairiuose kompiliatoriuose realizuojamos įvairiai, tad sunkiau pasiūlyti universalų mechanizmą jų lokalizavimui. Tačiau daugumoje kompiliatorių standartinės funkcijos žymimos vidiniais vardais, o tikrieji vardai atlieka tik leksikos elementų vaidmenį. Tokiais atvejais, jie gali būti internacionalizuoti gana lengvai.

Modifikatoriai savo paskirtimi ir realizacija yra labai panašūs į bazinius žodžius, todėl jų internacionalizavimas taip pat nesukelia problemų.

**Direktyvų vardai**. Direktyvų vardai paprastai nėra aprašomi programavimo kalbų standartuose. Direktyvos skirtos valdyti kompiliavimo procesą, tad jos yra gana susiję su konkretaus kompiliatoriaus realizacija, todėl jų vardų pasirinkimo teisė priklauso kompiliatoriaus gamintojui, ir paprastai direktyvos aprašomos kompiliatoriaus naudotojo dokumentacijoje.

**Leksikos elementų įkėlimas**. Kad leksikos elementus būtų galima visiškai lokalizuoti, internacionalizuojant kompiliatorių, turi būti numatytos dvi galimybės: juos pakeisti bei priskirti jiems sinonimus. Galimybė naudoti sinonimus, kai kuriais atvejais gali išspręsti teksto rinkimo problemas (pvz. surenkant matematinio ženklo, kurio nėra kompiuterio klaviatūroje, sinonimą).

Lokalizuotinus leksikos elementus laikant išorinėje duomenų bazėje gali būti išsprendžiama kompiliatoriaus bei jam sukurtų programų perkeliamumo problema (14 pav.). Įkeliant skirtingų lokalių leksikos elementus dinamiškai, kompiliatoriaus veikimo metu, jis taps nepriklausomas nuo konkrečios lokalės ir galės sukompiliuoti įvairioms lokalėms skirtas programas (pavyzdžiui, direktyvų arba kompiliatoriaus parinkčių pagalba nurodant programos lokalę).




*12 pav*. Leksikos elementų įkėlimas

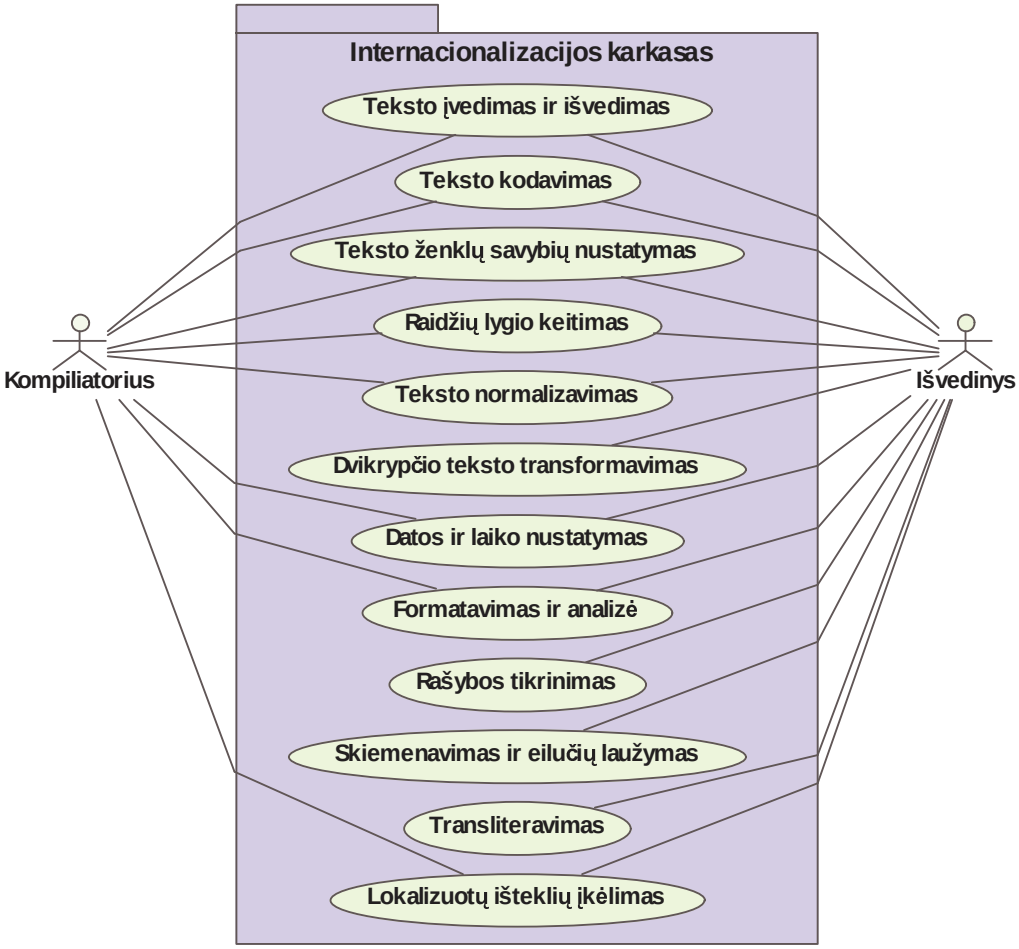
Suderinamumui su anksčiau sukurtomis programomis palaikyti tokiu atveju įmanoma įdiegti fiktyvią lokalę (14 paveikslėlyje ji sąlyginai pažymėta „en-ST“), į kurią būtų įtraukti leksikos elementai, atitinkantys standartinę kompiliatoriaus veikseną. Nustačius šią lokalę, būtų įmanoma sukompiliuoti anksčiau sukurtas programas, taip tarsi kompiliatorius sintaksė nebūtų išplėsta.

Suderinamumą su kitais tos pačios programavimo kalbos kompiliatoriais galima išlaikyti sukuriant preprocesorių verčiantį pirminį tekstą iš vienos lokalės į kitą. Jį sukurti nėra sudėtinga, nes sintaksiškai teisingos programos leksikos elementai į kitą lokalę verčiami vienareikšmiškai. Problemų gali iškils tik tuomet, kai programuotojo sukurti vardai (pvz. kintamųjų), sutaps su kitos lokalės leksikos elementų vardais. Tačiau, šias problemas taip pat nesunku išspręsti, pavyzdžiui automatiškai pakeičiant programuotojo sukurtus vardus pridedant papildomų ženklų.

Kultūrinių skirtumų, įtakojančių programavimo kalbos leksiką, yra daugiau nei pateikta šiame darbe. Projektuojant leksikos lokalės modelį, nederėtų joje aprašomų elementų aibę laikyti fiksuota, tam, kad ją vėliau būtų galima lengvai išplėsti. Lokalių aprašams tiktų naudoti atviras XML formatas. XML formato pranašumai, jo atvirumas ir tinkamumas medžio tipo duomenų struktūrų aprašymui. Todėl, jis sėkmingai pritaikomas sprendžiant panašius uždavinius, pavyzdžiui, aprašant lokales [Un05].

## 4.7. Internacionalizacijos karkasas

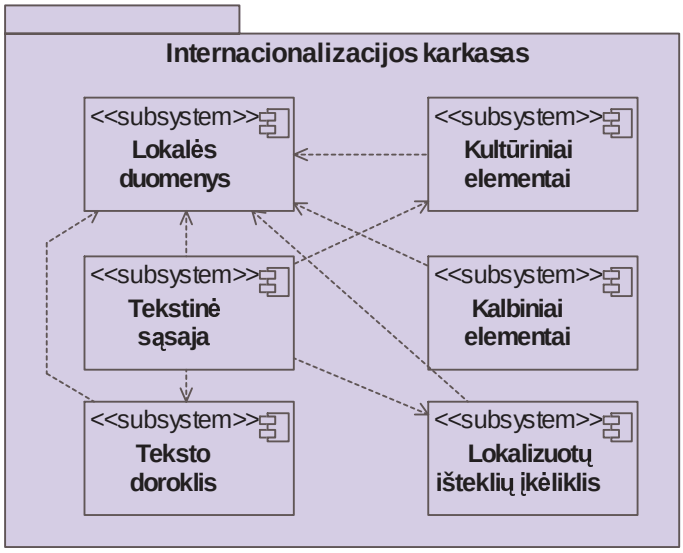
PĮ internacionalizavimas nėra lengvas uždavinys. Jį galima supaprastinti naudojant iš anksto parengtus PĮ kodo ruošinius, nes daugumą veiksmų ir duomenų naudojamų internacionalizuojant skirtingą PĮ galima panaudoti pakartotinai. Tokių ruošinių sistemą apibendrinsime naudodami internacionalizacijos karkasą. Pagrindiniai šio karkaso naudojimo atvejai pateikti 13 paveiksle




*13 pav*. Internacionalizacijos karkaso naudojimo atvejai

Internacionalizacijos karkasas yra naudojamas kompiliatoriaus ir jo sugeneruotų išvedinių. Nors kompiliatoriaus vykdymo meto biblioteka susieta su visomis internacionalizacijos karkaso paslaugomis, tačiau nelaikome kad juos naudoja kompiliatorius – laikoma, kad jos naudojamos išvedinio kai į jį patenka susieti vykdymo meto bibliotekos kodo fragmentai. Pavyzdžiui, kompiliatorius tiesiogiai nenaudojama dvikrypčio teksto transformavimo paslaugos, nes teksto kryptis reikšminga tik jo vaizdavimo metu kas nėra reikšminga kompiliatoriaus atveju.

Pagrindiniai internacionalizacijos karkaso komponentai ir ryšiai tarp jų pateikti 14 paveiksle. Tolesniame skyriuje jie apžvelgiami detaliau.




*14 pav.* Internacionalizacijos karkaso pagrindiniai komponentai

Šiuolaikinė PĮ paprastai veikia kooperuodama su platforma (OS, *.NET Framework*, vitualia Javos mašina ar pan.). Platforma valdo jos veikimą, o PĮ savo ruožtu naudoja platformos paslaugas, pavyzdžiui – failų sistemos, atminties, įvedimo-išvedimo, įrenginių valdymui ir kt. Šios paslaugos yra prieinamos per platformos API sąsają. Tai labai supaprastina PĮ kūrimą, nes paslaugas galima naudoti nesigilinant į jų vidinę sandarą ar veikimo smulkmenas (pvz. išvedant į ekraną tekstinį pranešimą, nereikia rūpintis ekrano raiška ar kitomis techninėmis galimybėmis), kas būtų labai sudėtinga ir neefektyvu. Šiuolaikinės platformos pateikia plačią paslaugų aibę, todėl PĮ kūrimas didžiąja dalimi susiveda į programos komponavimą panaudojant šias paslaugas [TDD95].

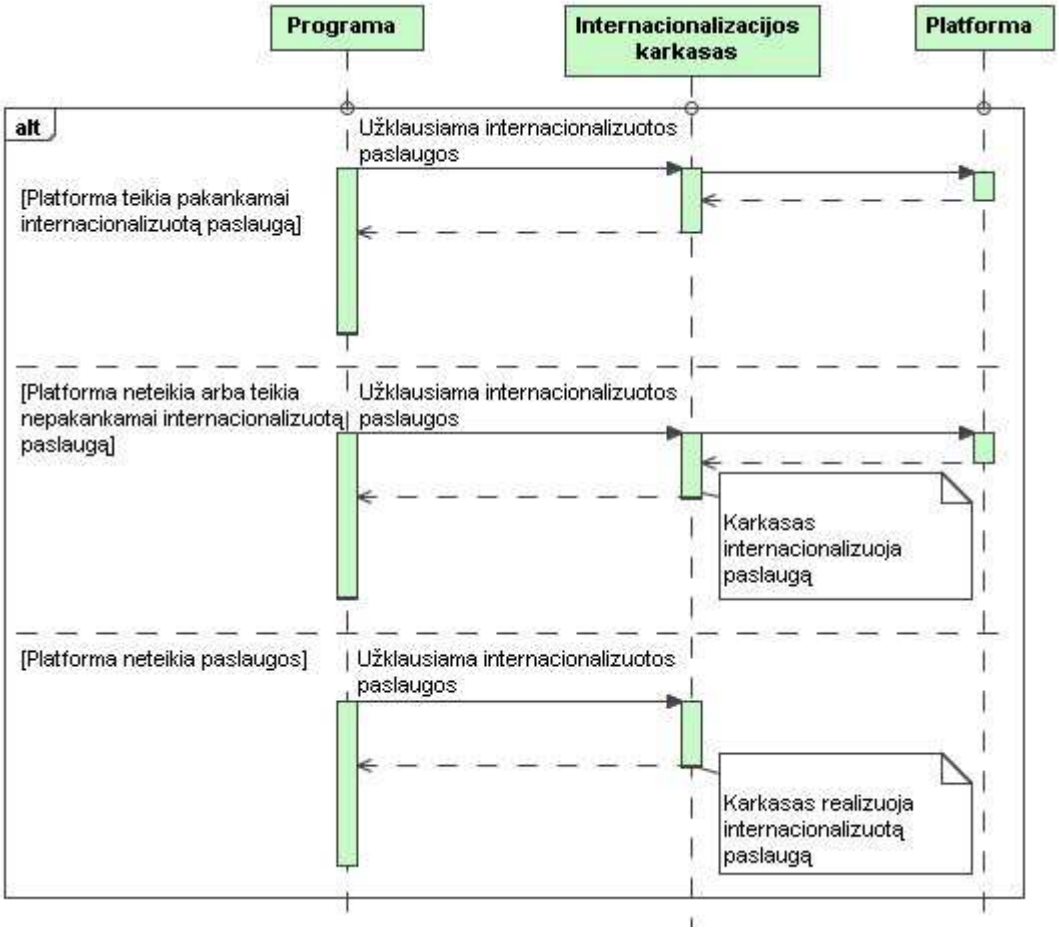
Paprastai PĮ nevaldo techninių įrenginių tiesiogiai naudodama jų tvarkykles, o atlieka tai panaudodama platformos paslaugas. Tokia praktika būtų neefektyvi ir sudėtinga ne tik programavimo, tačiau ir internacionalizacijos prasme, nes šiuolaikinės platformos paprastai pateikia jau pakankamai internacionalizuotas paslaugas, o pastaruoju atveju internacionalizavimu turėtų rūpintis pats programuotojas. Pavyzdžiui, naudojant seniau kurtą PĮ dar galima susidurti su tokia, kurioje tiesiogiai naudojama klaviatūros tvarkyklė, todėl naudojami tiesiogiai ir klaviatūros tvarkyklės gaunami klavišų kodai ir programa nereaguoja į OS nustatytą įvedimo metodą (kalbą) (dažniausiai tokiais atvejais įmanoma įvesti tik anglišką tekstą).

Internacionalizacijos karkaso sąveika su PĮ ir platforma yra pateikta 15 paveiksle.

Schemoje platformos ir internacionalizacijos karkaso paslaugos pavaizduotos tarpininko tarp žemo lygio paslaugų (techninės įrangos tvarkyklių arba tiesiog techninės įrangos) ir taikomosios PĮ vaidmenyje. Schemą galima išskirti į tris vertikaliai einančius lygmenis:

1. Vykdomoji programa naudoja jau internacionalizuotas platformos paslaugas.

   Vykdomoji programa naudoja internacionalizuotas platformos paslaugas ne tiesiogiai, o per tarpinę internacionalizacijos karkaso API sąsają. Tokiu būdu pasiekiama, kad PĮ nebūtų tiesiogiai priklausoma nuo platformos paslaugų ir tai padidina PĮ perkeliamumą įvairių platformų atžvilgiu. Nors pats internacionalizacijos karkasas nėra perkeliamas įvairių platformų atžvilgiu ir kiekvienai jų turi būti realizuotas atskirai.




*15 pav*. Internacionalizacijos karkaso sąveika su programa ir platforma

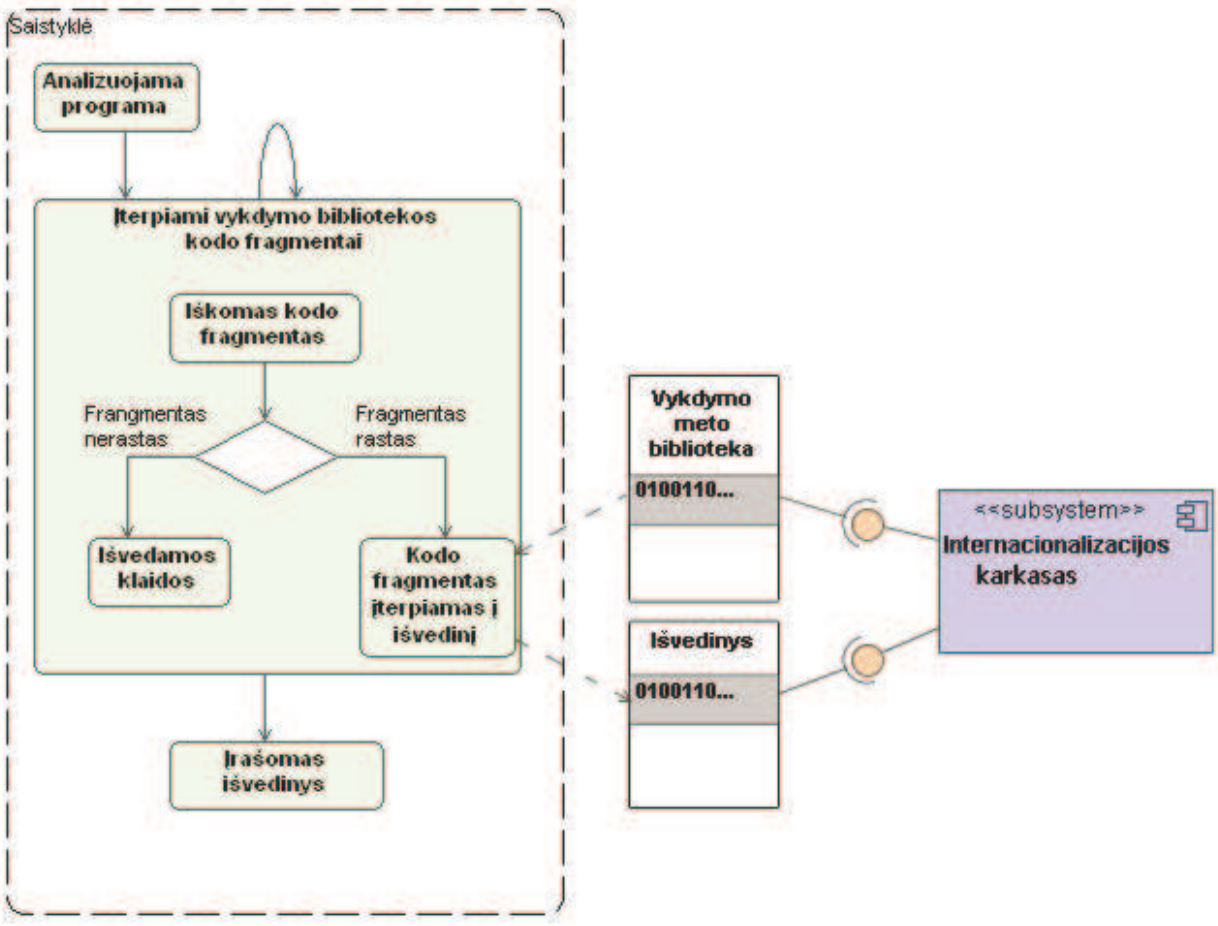
2. Platformos paslaugos nėra arba nepakankamai internacionalizuotos, todėl jų funkcionalumas internacionalizuojamas internacionalizacijos karkaso viduje.

Pavyzdžiui, *Windows 9x* OS didelė dalis paslaugų nėra pakankamai internacionalizuotos. Iš dalies šią problemą galima išspręsti naudojant priemonę *Microsoft Layer for Unicode*. Ši priemonė nepadaro OS paslaugų internacionalizuotomis, tačiau pasirūpina, kad internacionalizuotos PĮ, sukurtos naujesnės versijos *Windows* OS, funkcionalumas nebūtų prarastas jai veikiant *Windows 9x* aplinkoje [Ka01].

3. Žemo lygio paslaugos nėra realizuotos platformos, todėl jos turi būti internacionalizuotai realizuotos internacionalizacijos karkaso viduje.

Internacionalizacijos karkaso paslaugos yra prieinamos naudojant API. Tam, kad jos būtų prieinamos naudojant įvairias, o ne konkrečias programavimo sistemas, tikslinga API sukurti taip, kad ji nebūtų nuo jų priklausoma. Tai galima pasiekti naudojant šiuolaikines komponentinio programavimo technologijas (komponentai gali būti platinami ir panaudojami binariniame lygyje nepriklausomai nuo programavimo sistemos).

Tai taip pat palengvintų kompiliatoriaus vykdymo meto bibliotekos internacionalizavimą, nes joje saugomus kodo fragmentus būtų paprasčiau susieti su internacionalizacijos karkasu po to kai jie yra įkompiliuomi į išvedinį (16 pav.).




*16 pav.* Kodo fragmentų sąsaja su internacionalizacijos karkasu

Geras pavyzdys yra OpenOffice.org internacionalizacijos karkaso API sąsaja. Šios struktūros elementai (lokalės duomenys, kalendorius, gretintojas, ženklų klasifikatorius ir kt.) yra realizuoti atskirais UNO komponentais (UNO yra komponentinio objekto modelis, panašus į COM arba CORBA). Todėl išnaudojant šių komponentų sąsajų galimybes jie lengvai integruojami tarpusavyje ir į PĮ. Be to naudojant šią technologiją nesunku atnaujinti arba papildyti naujais komponentais ne tik internacionalizacijos karkasą, bet ir jau sukurtą PĮ [LD03] [LS01].

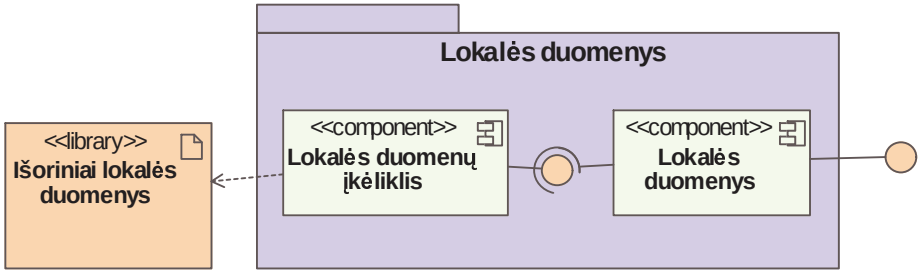
Blogas pavyzdys yra ICU (*International Components for Unicode*) projektas (http://icu.sourceforge.net/). Projekto tikslas sukurti PĮ internacionalizavimui skirtą karkasą sudarytą iš komponentų naudojančių Unikodo kodavimą. Tačiau iš tiesų ICU naudoja objektinę, o ne komponentinę technologiją, todėl ją sudarantys komponentai platinami ir į PĮ integruojami pirminio teksto lygyje, o API yra apribota dvejomis kalbomis: C++ ir Java. Programuojant kitomis kalbomis panaudoti šiuos komponentus beveik neįmanoma [IBM05].

## 4.8. Internacionalizacijos karkaso realizacija

### 4.8.1. Lokalės duomenys

Tai pagrindinis internacionalizacijos karkaso komponentas, nes nuo lokalės duomenų priklauso ir daugumos kitų komponentų veikimas. Nuoseklumui tarp įvairių platformų išlaikyti, tikslinga internacionalizacijos karkase naudoti savus lokalių duomenis. Todėl šis komponento paskirtis realizuoti lokalės duomenų įkėlimui ir apdorojimui skirtus algoritmus bei duomenų

struktūras. Išorinių lokalių duomenų aprašymui rekomenduotinas XML formatas, remiantis Unikodo techniniu standartu UTS 35 LDML (Locale Data Markup Language). Pradžiai galima panaudoti lokalių duomenis, remiantis šiuo standartu sukauptus duomenų bazėje CLDR (Common Locale Data Repository), vėliau juos papildant.




*17 pav.* Lokalės duomenų realizacija

### 4.8.2. Sąsaja su naudotoju

PĮ atlieta tarpininko vaidmenį tarp naudotojo ir kompiuterio, o PĮ sąsaja atlieka tarpininko vaidmenį tarp pačios PĮ ir vartotojo. Ši PĮ dalis daugiausia atsakinga už informacijos įvedimą ir pateikimą vartotojui. Ją internacionalizuojant siekiama, kad tai būtų galima atlikti teisingai pagal vartotojo lokalėje galiojančias nuostatas arba kad tokias galimybes būtų galima sudaryti lokalizavus PĮ. Įvedamos arba pateikiamos informacijos pobūdis gali būti įvairus: tekstas, grafika, garsas ir kt. Informacijos įvedimui gali būti naudojamos įvairios priemonės – klaviatūra, pelė, jutiklinis kilimėlis ar ekranas, mikrofonas ir kt.; pateikimui – ekranas, garso kolonėlės ir kt. Sąsaja su naudotoju apima įvairius PĮ elementus: įvedimo metodus, pranešimų katalogus, langų geometriją, spalvas, piktogramas, simbolius, garsus ir pan.

Pagrindinį įvedamos ir išvedamos informacijos srautą daugumoje PĮ sudaro tekstinė informacija. Todėl galimybių teisingai ją įvesti ir išvesti sudarymas yra vienas iš svarbiausių uždavinių kuriant PĮ. Deja, PĮ gamintojai dažnai neatsižvelgia arba nepakankamai atsižvelgia į tai, kad nepakankamai internacionalizuota PĮ gali nepalaikyti kitose kultūrose naudojamų raštų. T.y. nesudaro papildomų galimybių būtinų teisingam šių raštų teksto įvedimui ir išvedimui (tai labiausiai būdinga anglakalbiams gamintojams, nes jų raštas šiuo aspektu yra vienas iš paprasčiausių. PĮ leidžia jį teisingai įvesti ir išvesti, ne jokio papildomo pritaikymo).

- Teksto įvedimas

  Teksto įvedimo problematiškumas priklauso nuo rašte esančių raidžių kiekio ir jų paskirties. Alfabetuose, abdžaduose raidžių kiekis yra pakankamai nedidelis (~30) tam, kad jas būtų galima sutalpinti ant atskirų standartinės klaviatūras klavišų. Kai kuriais atvejais kai norima, kad ta pačia klaviatūros tvarkykle būtų galima įvesti ne vieno, o kelių raštų raides gali būti naudojamas trečiojo lygio klavišas arba papildoma klavišų išdėstymo grupė. Pavyzdžiui, taip lietuviška standartinė klaviatūros tvarkyklė (LST 1582) naudoja trečio lygio ženklus, nes į ją be lietuviškų raidžių taip pat yra įtrauktos ir angliškos (x, w, q) bei kiti dažnai naudojami ženklai, kuriems nepakanka dviejų lygių.

  Dalis abugidų turi daugiau raidžių nei skiriama joms klavišų standartinėse klaviatūrose (pavyzdžiui, Devanagari jų turi 48). Tai nesukelia didelių įvedimo problemų, greičiau nepatogumą, nes perteklinių raidžių įvedimo problema sprendžiama naudojant antrąjį klaviatūros lygį[28] [HH97].

[28] Abugidai turi tokią galimybę, nes juose raidės neskirstomos į didžiąsias ir mažąsias

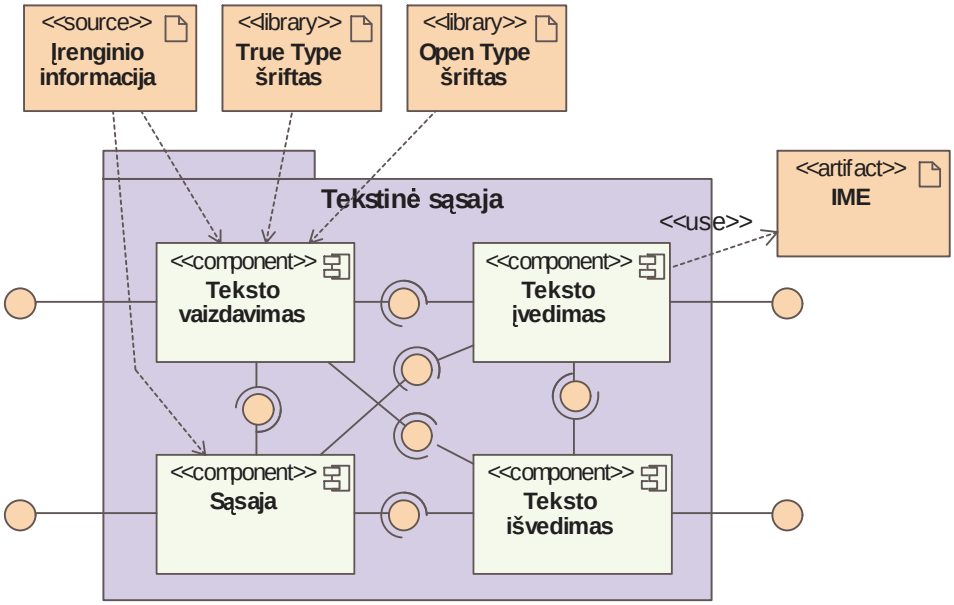
Problematiškiausi teksto įvedimo požiūriu yra skiemeniniai ir logo-skiemeniniai raštai, nes juose naudojamų raidžių kiekis gali kelis ir daugiau kartų viršyti standartinėje klaviatūroje esančių klavišų kiekį. Šių raštų įvedimo metodai dar vadinami kompleksiniais. Skiemeniniuose raštuose raides galima vienareikšmiškai atpažinti pagal jų fonetinę sandarą (jas sudarančių fonemų dažniausiai išskiriama ~30), todėl jų įvedimui daugiausia naudojami fonetiniai įvedimo metodai. Sudėtingesnė situacija su logo-skiemeniniais raštais, nes juose raidės gali turėti keletą skirtingų tarimų arba kelios ir daugiau raidžių gali turėti tą patį tarimą. Tokiais atvejais nepakanka vien įvesti raidę pagal jos fonetinę sandarą. Jų įvedimo metodas dar turi suteikti galimybę pasirinkti teisingą raidę iš galimų raidžių sąrašo. Panašią pasirinkimo galimybę turi suteikti ir kiti logo-skiemeninių raštų įvedimo metodai: struktūrinis ir rašysenos. Struktūrinis metodas remiasi raidžių išskaidymu į tipines struktūrines dalis pagal jų vaizdą (šis išskaidymas neleidžia vienareikšmiškai atgaminti raidės iš jos struktūrinių dalių). Rašysenos metodas remiasi rašysenos atpažinimo priemonėmis (tai nebūtinai išorinės įvedimo priemonės, raidės vaizdas gali būti piešiamas pele specialiame įvedimo metodo lange) (vienareikšmiškai raidžių atpažinti negalima dėl žmogiškų įvedimo netikslumų).

- Teksto vaizdavimas

  Teksto vaizdavimo sudėtingumas, kitaip nei įvedimo, priklauso nuo rašto sudėtingumo ir krypties. Sudėtingiausi raštai, kuriuose ženklai gali keisti formą ir netgi ženklų eiliškumą priklausomai nuo konteksto (šalia esančių ženklų). Šiuo aspektu sudėtingiausia yra abugidų raštų grupė. Pavyzdžiui, Devanagari rašte jungiant raides gaunama daugybė naujų raidžių, kurios nėra įtrauktos į Unikodo standartą. Šriftuose šioms raidėms skirti papildomi glifai užimtų daug vietos, todėl paprastai jų vaizdas ekrane yra sukuriamas dinamiškai. Tam naudojamos šių raidžių atskiros struktūrinės dalys (jų glifams koduoti naudojami kompleksiniai šriftai, pavyzdžiui, OpenType) ir jų komponavimo taisyklės. Apibendrintai šias taisykles apibrėžia Unikodo standartas [Un03], detaliau jų realizacija nagrinėjama OpenType šriftų specifikacijoje [Mi04] ir kituose šaltiniuose.

  Kituose raštuose tokių sudėtingų, raidžių vaizdavimo taisyklių kaip abugiduose nėra. Jų pagrindinės raidės apibrėžtos Unikodo standarto, todėl jų vaizdavimas nesudėtingas. Problemų iškyla tik tada kai yra naudojami specialūs diakritiniai ženklai, pavyzdžiui, kirčių žymėjimui (jie naudojami ir lietuvių rašte). Tokiais atvejais raidės su šiais ženklais Unikodo standarte gali būti neapibrėžtos ir joms gauti gali tekti naudoti kombinacinius diakritinius ženklus. Tačiau raidžių su kombinaciniais diakritiniais ženklais vaizdavimas nėra pakankamai gerai apibrėžtas, todėl atskirais atvejai gali reikėti atitinkamo PĮ pritaikymo, kad jos būtų teisingai vaizduojamos [Un03].

Tais atvejais kai platforma nepateikia pakankamai internacionalizuotų teksto įvedimo ir išvedimo paslaugų, tikslinga naudoti tekstinę sąsają realizuotą internacionalizacijos karkaso viduje. 18 paveiksle pateikta apibendrinta tokios tekstinės sąsajos bei teksto įvedimo ir išvedimo paslaugų realizacija.



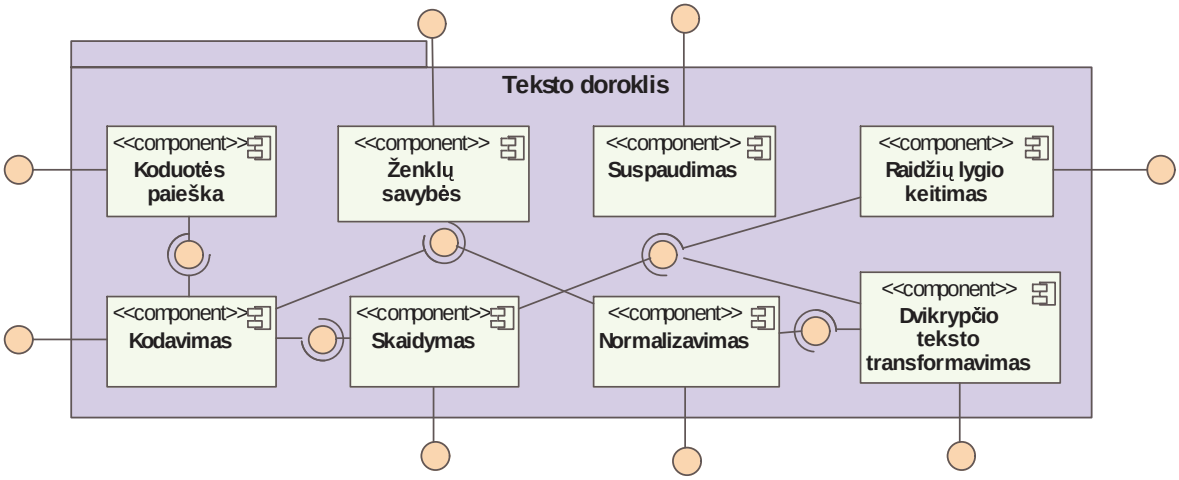

*18 pav.* Tekstinės sąsajos realizacija

Teksto srautai šios sąsajos pagalba įvedami ir išvedami lange, kurį realizuoja „Sąsajos“ komponentas. Šį komponentą naudoja ir visi kiti tekstinę sąsają sudarantys komponentai.

Teksto įvedimo komponentas, įvedimui naudoja platformos arba kitą išorinį įvedimo metodo redaktorių. Teksto vaizdavimo komponentas naudoja informaciją apie įrenginį į kurį jis bus išvedamas (pavyzdžiui, teksto vaizdavimui svarbu, kokia ekrano raiška, kiek ekrane yra taškų ir pan.) bei šriftus. True Type šriftai gali būti naudojami paprastiems raštams, kurių ženklai įtraukti į Unikodo standartą ir jiems galima priskirti atskirus šrifto glifus, išvesti. Sudėtingiems raštams yra naudojami Open Type šriftai, nes jų ženklų glifų komponavimui naudojamos šiuose šriftuose saugomos struktūrinės dalys ir jų išdėstimo scenarijai.

### 4.8.3. Teksto doroklis

Realizuoja duomenų struktūras ir algoritmus skirtus tekstui užkoduoti, saugoti ir apdoroti. 19 paveiksle pateikta apibendrinta teksto doroklį sudarančių komponentų ir ryšių tarp jų schema. Panagrinėsime kiekvieną iš jų detaliau.



*19 pav.* Teksto doroklio realizacija

- Kodavimas

  Realizuoja teksto kodavimą ir perkodavimą į įvairias kodų lenteles (plačiau apie kodavimą rašoma sekančiame 4.3 skyrelyje). Dauguma platformų turi pakankamai plačias ženklų konvertavimo galimybes, tačiau netgi tarptautiniais standartais priimtos kodų lenteles ne visose platformose realizuojamos vienodai (be išimčių). Pavyzdžiui, Shift-JIS koduotės ženklas 5C16 gali būti suprantamas kaip pasviras brūkšnys *Windows* OS ir kaip Jenos (Japonijos piniginio vieneto) simbolis Unix OS. Kodavimo ir perkodavimo realizacija internacionalizacijos karkaso viduje leistų išspręsti šią ir kitas su konvertavimu susijusias problemas (UTS 22).

- Duomenų tipai.

  Teksto doroklis turi realizuoti eilučių ir simbolių tipus skirtus Unikodu koduotam tekstui. Jų realizacija turi būti suderinama su UTF-8, UTF-16, UTF-32 Unikodo kodavimo metodais. Turi būti realizuotas eilučių konvertavimas ir vieno tipo į kitą [Un03].

- Ženklų savybės.

  Unikodas standartas ne tik apibrėžia ženklų aibę, tačiau išsamiai apibrėžia ir kiekvieno atskiro ženklo savybes, bei pateikia pastabas susijusias su jų paskirtimi ir naudojimu. Šio komponento realizacija turėtų atitikti UTR 23, UAX 24 dokumentus.

- Raidžių lygio keitimas

  Egzistuoja trijų tipų raidės: raidės neturinčios lygių, turinčios viršutinį ir apatinį lygius (didžiosios, mažosios), bei raidės be šių dviejų lygių turinčios dar ir trečią, antraštinį lygį (pavyzdžiui DŽ (U+01C4), dž (U+01C6), Dž (U+01C5)). Raidžių lygio keitimas gali būti atliekamas dvejopai: bendruoju ir nuo lokalės priklausomu būdu. Pirmuoju atveju keitimas atliekamas pagal Unikode apibrėžtas bendras taisykles, antruoju atveju pagal lokalėje galiojančias taisykles. Pavyzdžiui, bendruoju atveju „i“ raidės viršutinio lygio atitikmuo yra „I“, o tuo tarpu turkų lokalėje „İ“ (U+0130)(su tašku). Taip pat egzistuoja ir kitokių komplikuotų situacijų susijusių su raidžių lygio keitimu, į kurias turi būti atsižvelgta realizuojant šį komponentą [Un03] (UAX 21[29]).

- Skaidymas

  Bendriausi veiksmai atliekami skaidant tekstą yra jo skaidymas į grafemas, žodžius ir sakinius. Čia atskiriama ženklo ir grafemos sąvokos, nes grafemos gali būti sudarytos iš daugiau nei vieno Unikodo ženklo (nepaisant to, kad vartotojas matydamas grafemą ekrane identifikuoja kaip vieną ženklą). Pavyzdžiui, ženklų a (U+0061) ir ◌̨ (U+0328) seka yra vaizduojama kaip vienas ženklas „ą“. Teksto skaidymas į žodžius ir sakinius yra dar sudėtingesnis, nes skyrybos ženklai įvairiose lokalėse ir netgi toje pačioje lokalėje priklausomai nuo rašybos taisyklių atlieka nevienareikšmį vaidmenį (UTR 17).

- Normalizavimas

  Panaudojant kombinacinius ženklus dalis Unikodo ženklų gali būti koduojami daugiau nei vienu būdu. Remiantis Unikodo standartu visi taip gaunami to paties ženklo kodai yra laikomi ekvivalenčiais. Normalizuojant atliekamas tekste esančių ženklų kodavimo būdų suvienodinimas (UAX 15).

[29] Nuo 4.0 Unikodo versijos įtrauktas į standartą nebe kaip priedas, o kaip standarto dalis.

- Dvikrypčio teksto transformavimas

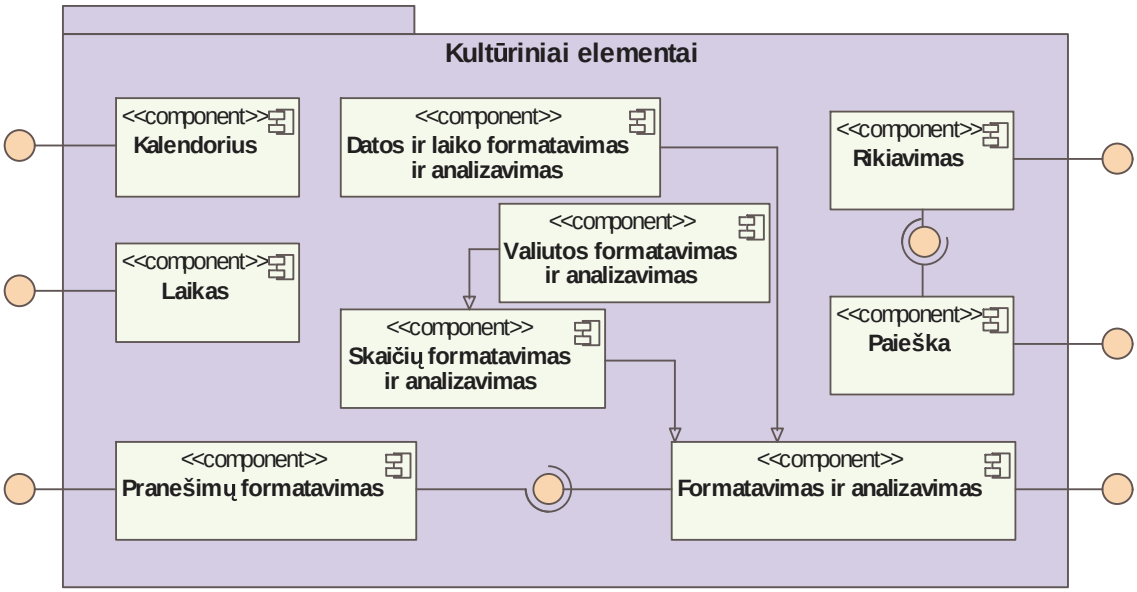
  Dvikryptį tekstą didžiąją dalimi sudaro tekstas rašomas iš dešinės į kairę (tokias rašto sistemas daugiausia naudoja viduriniųjų rytų kalbos: arabų, urdu, hebrajų ir kt.) su intarpais rašomais iš kairės į dešinę. Tokiais intarpais gali būti skaičiai arba kitų raštų tekstas. Įrašant į laikmeną teksto dvikryptiškumas yra panaikinamas, t.y. teksto ženklai yra įrašomi tokia tvarka kokia jie yra skaitomi – logine tvarka. Ženklų išdėstymo tvarka, kuria jie yra vaizduojami vadinama vaizdavimo tvarka. Dvikrypčio teksto transformatorius kaip tik ir skirtas keisti loginę ženklų išdėstymo tvarką į vaizdavimo ir atvirkščiai (UAX 9) [AS01].

- Suspaudimas

  Tam kad Unikodu koduotas tekstas galėtų užimti mažiau vietos Unikodo standartas pateikia metodą jo suspaudimui (UTR 6).

### 4.8.4. Kultūriniai elementai

Realizuoja algoritmus tiesiogiai susijusius su lokalės duomenimis. 20 paveiksle pateikta apibendrinta kultūrinius elementus sudarančių komponentų ir ryšių tarp jų schema. Panagrinėsime juos detaliau.




*20 pav.* Kultūrinių elementų realizacija

- Kalendorius ir laiko matavimas

  Pasaulyje yra naudojami įvairūs kalendoriai, išsamiai juos aprašo Dershowitz ir Reingold [DR01]. Plačiausiai pasaulyje yra naudojamas Grigaliaus kalendorius. Tačiau net ir jis skirtingose lokalėse turi gali būti skirtingai naudojamas. Pavyzdžiui, gali skirtis pirma savaitės diena (JAV pirma savaitės diena yra sekmadienis, Lietuvoje pirmadienis), švenčių dienos. Šis komponentas realizuoja datų perskaičiavimo iš vieno kalendoriaus į kitą funkcijas.

  Įprastą laiko matavimą įtakoja laiko juosta (identifikuojama valandų nuokrypiu nuo Grinvičo laiko). Šis nuokrypis susijęs ne tik su lokale (per lokalės regioną gali eiti ir kelios laiko juostos), bet yra priklausomas ir nuo geografinės padėties. Be to jis gali kisti

metų bėgyje priklausomai nuo to ar yra naudojamas vasaros laikas (regionai esantys arčiau ekvatoriaus dažniausiai jo nenaudoja).

Universalus datų ir laiko žymėjimas, kuris šiuo atveju gali būti naudojamas vidinei datų ir laiko realizacijai, bei informacijos apsikeitimui, apibrėžiamas standartu ISO 8601. Naudingų pastabų susijusiu su jo realizacija pateikiama RFC 3339 dokumente.

- Formatavimas ir analizė

  Formatuojant universaliu, skaitmeniniu formatu išreikšta informacija paverčiama į žmogui suprantamą tekstinį formatą atitinkantį jo kultūrines nuostatas, o analizuojant atliekamas atvirkščias veiksmas. Tiek formatuojant, tiek analizuojant informaciją naudojami lokalėse apibrėžiami skaičių, valiutos, datų, laiko ir kitų elementų formatai. Šis komponentas taip pat teikia ir pranešimų formatavimo paslaugas, kai į pranešimą įterpiami nuo lokalės priklausomi elementai arba atliekamas gramatinis žodžių formų kaitymas.

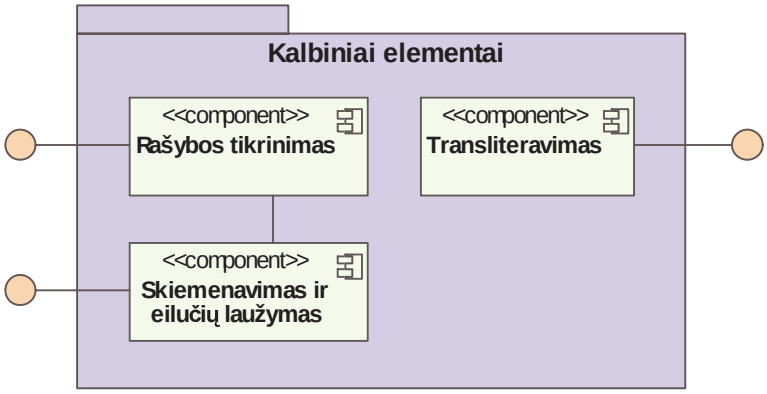
  Internacionalizuojant programą derėtų vengti tiesiogiai sujungtų pranešimų, panaudojant sudėties operaciją arba eilučių jungimo funkcijas. Tam derėtų naudoti formatavimo funkcijas, kurios sujungia arba įterpia eilutes parametrų pagalba. Lokalizuotame tekste gali keistis žodžių išsidėstymo tvarka, todėl toks būdas leistų išvengti galimų su tuo susijusių vertimo problemų.

- Rikiavimas ir paieška

  Rikiavimas labai svarbi ir PĮ dažnai naudojama funkcija. Rikiavimas glaudžiai susijęs su paieška, nes surikiuotų elementų aibėje paieška gali būti atliekama žymiai sparčiau. Rikiavimą daro sudėtingesnį tai, kad skirtingose lokalėse jis gali būti apibrėžtas skirtingai. Taip pat ir toje pačioje lokalėje gali būti naudojami keli rikiavimo metodai skirtingiems elementams rikiuoti (pavyzdžiui, žodynų elementai daugelyje lokalių yra rikiuojami kitaip nei telefonų knygų elementai). Neabėcėlinių raštų (pavyzdžiui, Rytų Azijos ideografinių raštų) elementų rikiavimas gali būti fonetinis arba paremtas ženklų vaizdavimo savybėmis. Ženklų rikiavimą taip pat apsunkina ir tai, kad dalis ženklų gali turėti daugiau nei vieną kanoniškai ekvivalenčią formą (pavyzdžiui, kai naudojamos kombinacinės sekos), o taip pat jų rikiavimas gali priklausyti nuo konteksto (pavyzdžiui, kai kuriose lokalėse H < Z, tačiau CH > CZ). Unikodo techninis standartas UTS 10 apibrėžia apibendrintą penkių pakopų rikiavimo algoritmą. Šiose pakopose yra lyginami 1) baziniai ženklai; 2) diakritiniai ženklai; 3) raidžių lygis; 4) skyrybos ženklai; 5) kombinaciniai ženklai. Tai apibendrintas algoritmas, skirtingose lokalėse arba atskirais atvejais būti naudojamos ne visos paminėtos rikiavimo pakopos arba gali skirtis jų eiliškumas. Panaši metodika kaip ir rikiavimo atveju taikoma ir paieškos algoritmuose. Paieškos atveju ne visos pakopos gali būti reikšmingos, todėl jos gali būti praleidžiamos, pavyzdžiui jei nereikšmingas raidžių lygis tuomet jis praleidžiama trečia pakopa (UTS 10).

### 4.8.5. Kalbiniai elementai

Realizuoja rašybos tikrinimo, skiemenavimo, eilučių laužymo ir kt. su kalba susijusius elementus. 21 paveiksle pateikta apibendrinta kalbinius elementus sudarančių komponentų schema. Panagrinėsime juos detaliau.




*21 pav.* Kalbinių elementų realizacija

- Rašybos tikrinimas

  Rašybos tikrinimas yra labai plačiai pritaikoma galimybė. Ji gali būti naudojama ne tik rengiant dokumentus, tačiau ir kalbantis pokalbių programomis, tinklalapiuose pildant anketas ir pan. Rašybos taisyklės skirtingose kalbose ir raštuose stipriai skiriasi ir jos yra gana sudėtingos realizacijos prasme. Paprastai šalia formalių taisyklių daugelyje kalbų egzistuoja dar ir daugybė išimčių, kada šios taisyklės nėra taikomos, kas rašybos tikrinimą apsunkina dar labiau. Trumpai pagrindinius rašybos tikrinimo realizacijos kompiuteryje metodus apžvelgia Mitton [Mi96].

  Realizuojant rašybos tikrinimą verta atkreipti dėmesį į atvirąsias[30] rašybos tikrinimo priemones. Jos yra sparčiai tobulinamos ir yra nemokamos. Be to jas leidžiama plėtoti bei taikyti įvairiais tikslais. Pavyzdžiui apie jų pritaikymą lietuvių kalbos rašybos tikrinimui rašo Agejevas [Ag02].

- Skiemenavimas, eilučių laužymas

  Skiemenavimas glaudžiai susijęs su eilučių laužymu, nes dažniausiai jis ir naudojamas tam, kad surasti optimaliausias eilučių lūžio vietas. Skiemenavimas taip pat iš dalies susijęs su rašybos tikrinimo komponentu, nes žodžių skaidymui į skiemenis gali būti naudojami tie patys gramatikos ir morfologinės analizės algoritmai kaip ir rašybos tikrinimo atveju.

  Eilutės lūžio vieta nustatoma dviem etapais: 1) nustatomos gali mos eilutės lūžio vietos (jas gali sąlygoti į eilutę įterpti valdymo ženklai žymintys galimas lūžio vietas (pvz. U+000D, U+2028 ir kt.) arba jas draudžiantys (pvz. U+00A0 ir kt.)); 2) teksto formatavimo algoritmas parenka optimaliausią eilutės lūžio vietą. Įvairi PĮ gali eilučių laužymo algoritmus realizuoti įvairiai. Unikodo standartas pateikia apibendrinto, tinkamo daugumai atvejų eilučių laužymo algoritmo specifikaciją [Un03] (UAX 14). Naudingų pastabų susijusių su dinaminio ir tikimybinio programavimo metodų taikymu eilučių laužymo algoritmų realizacijai pateikia [Fi00] [Bo03].

- Transliteravimas

  Transliteruojant ženklai iš vienos rašto sistemos pakeičiami į kitą pagal jų fonetinį atitikimą. Pavyzdžiui, rusiškas tekstas „биологическом“ transliteravus kirilicą į angliškus rašmenis gali būti rašomas kaip „biologichyeskom“. Geresnės transliteravimo

[30] (Angl. *open source*). Pagal kompiuterinės leksikos aiškinamąjį žodyną (www.likit.lt).

sistemos ne tik pakeičia tekstą iš vienos rašto sistemos į kitą pagal fonetinį ženklų atitikimą, tačiau atsižvelgdamos ir į žodžių tarimą [KG97].

### 4.8.6. Lokalizuotų išteklių įkėliklis

Lokalizuojamus išteklius sudaro naudotojo sąsajos langų šablonai (geometrija), pranešimų tekstai, paveikslai, piktogramos ir daugelis kitų nuo lokalės priklausomų PĮ elementų. Šiuo metu jų atskyrimui nuo programos kodo naudojama keletas metodų: GetText (remiasi GNU GetText priemonių rinkinio taikymu, daugiausia naudojamas internacionalizuojant atvirąsias programas), *.RESX* (ištekliai aprašomi XML kalba paremtu formatu, naudojamas *.NET* platformose), *.RC* (ištekliai įkompiliuojami į vykdomuosius failus arba dinamines bibliotekas, daugiausia naudojamas internacionalizuojant *Windows* programas), ResourceBundles (sąsajos elementai ir tekstai atskiriami į tekstinius failus, naudojamas programuojant *Java*). Visi šie metodai turi ir gerų savybių ir trūkumų. Plačiau jie aprašyti 4.4 skyrelyje. Internacionalizacijos karkase išteklių valdymas gali būti realizuojamas remiantis vienu iš šių metodų arba sukuriant naują metodą, kuris apimtų gerąsias jų savybes ir neturėtų rimtų trūkumų.

# V. KOMPILIATORIAUS INTERNACIONALIZAVIMO EKSPERIMENTAS

## 5.1. Eksperimento tikslai

- Vienas iš pagrindinių šio eksperimento tikslų surinkti daugiau praktinių žinių susijusių su kompiliatorių internacionalizavimu.
- Pademonstruoti darbe pateiktų žinių praktinį pritaikymą.
- Internacionalizuoti *Free Pascal* kompiliatorių. Šis kompiliatorius naudojamas Lietuvos mokyklose mokymo tikslais, todėl jo internacionalizavimas ir lokalizavimas atneštų pedagoginės naudos.

## 5.2. Eksperimento objektas

PĮ internacionalizavimas yra jos kūrimo proceso dalis ir jam atlikti būtina prieiga prie pirminių jos tekstų, todėl buvo pasirinktas nemokamas, atvirojo teksto kompiliatorius. Be to atvirosiose programose dažnai būna daugiau internacionalizacijos problemų, negu komercinėse.

*Free Pascal* kompiliatoriaus pasirinkimą taip pat lėmė tai, kad tai vienas populiariausių Pascal kalbos kompiliatorių pasaulyje: daugelyje šalių jis naudojamas mokymo tikslais (šiuo metu jis naudojamas mokymui ir Lietuvos mokyklose), taip pat jis naudojamas rengiant tarptautines informatikos olimpiadas. Mokymui skirtoms programoms yra keliami didesni internacionalizacijos reikalavimai nei įprastoms programoms. Be to *Free Pascal* kompiliatoriaus pritaikymo Lietuvos mokykloms [DL01] [La01] patirtis atskleidė, kad jis turi daugybę internacionalizacijos trūkumų, kurie ženkliai apsunkina jo lokalizavimą. Vadovaujantis 3.5 skyrelyje pateiktais teiginiais liečiančiais kompiliavimo meto internacionalizaciją, galimos dvi šio kompiliatoriaus lokalizavimo strategijos:

1. Lokalizuoti tiesiogiai, nekeičiant esamos internacionalizacijos.
2. Pakeisti internacionalizaciją į susaistymo arba vykdymo meto internacionalizaciją.

Nors pirmoji strategija leidžia greičiau gauti lokalizuotą produktą ir pradžioje reikalauja mažesnių investicijų, tačiau antrosios strategijos metu įdėtos pradinės investicijos turi ilgalaikį efektą ir atsiperka bei atneša naudą lokalizuojant tolesnes PĮ versijas. Todėl buvo pasirinkta ne tik lokalizuoti *Free Pascal* kompiliatorių, bet ir internacionalizuoti, kas sutapo ir su šio darbo uždaviniais.

Ištyrus *Free Pascal* kompiliatoriaus internacionalizaciją 3.2 skyriuje pateiktu metodu buvo rasti ir susisteminti jo internacionalizacijos trūkumai. Tyrimo rezultatai pateikti 6 priede.

Visiškai pašalinti pastebėtus trūkumus nebuvo įmanoma dėl laiko ir išteklių stokos, todėl buvo pasirinkta pašalinti tik svarbiausius iš jų, t.y. tuos, kurie kelia daugiausia problemų adaptuojant *Free Pascal* kompiliatorių mokykliniam programavimo kursui.

Mokymui skirtiems kompiliatoriams keliami reikalavimai skiriasi nuo profesiniam darbui skirtiems kompiliatoriams keliamų reikalavimų [La06]. Tai lemia ir jų internacionalizacijos savybėms teikiamų prioritetų paskirstymą.

Dauguma specialistų sutaria, kad pagrindiniai mokyklinio programavimo kurso tikslai perteikti programavimo sampratą, kad mokinys suvoktų bendruosius programavimo principus, praktinę algoritmų ir programavimo naudą – tai padėtų moksleiviams geriau apsispręsti dėl tolesnio specializavimosi informatikos srityje. Programavimo kurso metu labiau siekiama lavinti ne praktinius programavimo įgūdžius, bet kūrybiškumą, fantaziją, ugdyti tvarkingą, nuoseklų,

loginį mąstymą [Da05]. Efektyviausiai siekti šių tikslų galima pasitelkiant kuo paprastesnės priemonės, kad mokiniai galėtų jas lengviau perprasti ir išmokti naudoti. Todėl mokymui naudojama tik labai ribota programavimo kalbos ir kompiliatoriaus galimybių dalis.

Pagrindiniai keliami uždaviniai lokalizuojant mokykliniam kursui skirtą PĮ yra: išversti jos tekstus, adaptuoti nuo lokalės priklausomus elementus ir išversti žinynus. Lokalizuojant kompiliatorių papildomai prisideda dar keli uždaviniai: adaptuoti programavimo kalbos leksiką, suderinti su lokalės nuostatomis vykdymo meto biblioteką, tiek kiek ji yra naudojama mokykliniam kursui – pavyzdžiui, vienas iš svarbiausių uždavinių: kad sukompiliuotos programos leistų teisingai (lokalės atžvilgiu) įvesti ir išvesti duomenis. Kompiliatoriaus internacionalizavimo uždaviniai sudaryti galimybes tam atlikti lengvai ir nekeičiant pirminio jo teksto.

Internacionalizuojant *Free Pascal* kompiliatorių buvo išskirti keli etapai, kurių metu internacionalizuotus elementus galima suskirstyti į šias grupes:

1. Duomenų kodavimas.
2. Įvedimas ir išvedimas.
3. Leksikos elementai.
4. Lokalės elementai.
5. Išteklių atskyrimas.
6. Platformos paslaugos.

Atkreiptinas dėmesys į tai, kad kai kurie etapai negali būti vykdomi anksčiau už kitus. Pavyzdžiui, negalima neįdiegus teisingo duomenų kodavimo vykdyti kitų etapų, nes jie yra nuo jo priklausomi.

## 5.3. „*FPS*“ programavimo terpė

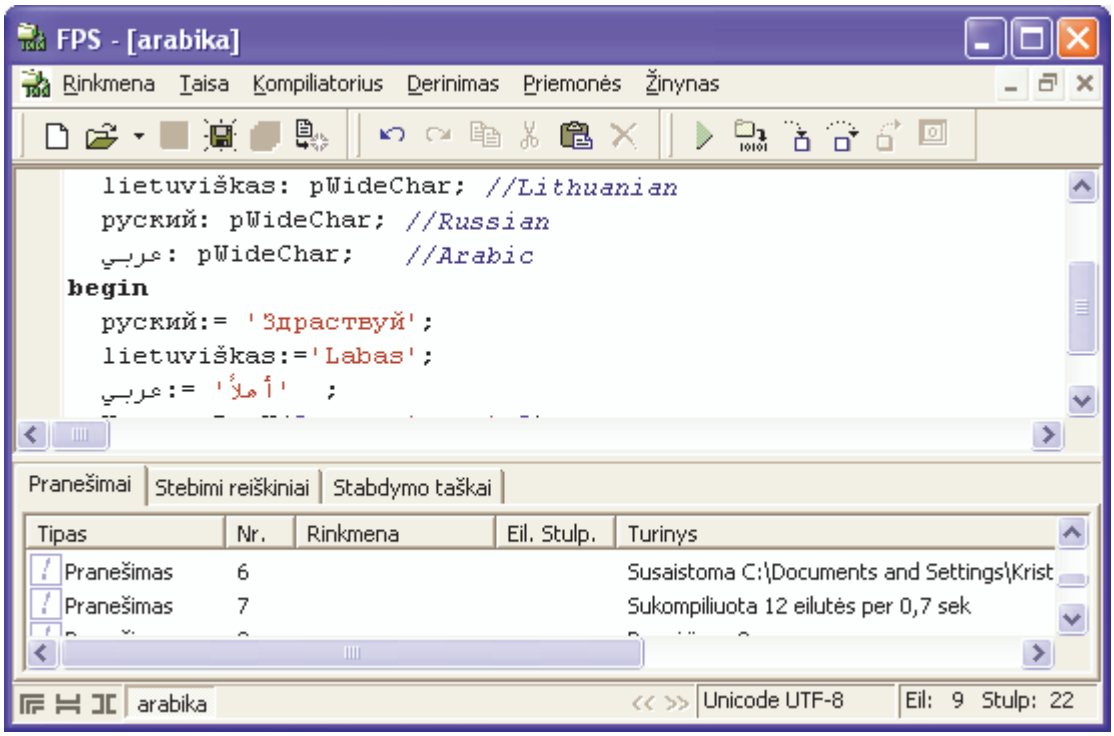
Pagrindinis *Free Pascal* trūkumas pastebėtas vertinant jo pritaikymo mokykliniam kursui galimybes: tinkamos programavimo terpės nebuvimas. Kartu su juo platinama programavimo terpė *FP* veikia tekstine veiksena ir nėra pakankamai stabili. Todėl buvo nuspręsta programavimo terpę, kuri atitiktų iš anksto iškeltus kriterijus, sukurti patiems.

Naujai sukurta programavimo terpė pavadinta *FPS* (17 pav.). Siekiant, kad ją būtų paprasta įdiegti ir naudoti, ji buvo integruota su *Free Pascal* kompiliatoriumi į vieną paketą. Iliustruoti kompiliatoriaus išplėtimams priede pateikiamas kompiliatoriaus išplėtimų tekstas gautas lyginant jo pirminį tekstą su originaliu pirminiu tekstu naudojant priemonę *GNU Diff*.

Naujoji programavimo sistema – *FPS* jau sėkmingai taikoma mokykliniame informatikos kurse [La06].

*FPS* programavimo terpė sukurta remiantis iš anksto iškeltais kriterijais, lemiančiais ir jos metodinį tinkamumą programavimo kursui: **paprastumas** – terpė turi būti paprasta naudotis ir išmokti, ji turi būti neperkrauta mokykliniam kursui nereikalingomis galimybėmis, tam, kad mokiniai kuo mažiau laiko praleistų mokydamiesi ja naudotis; **panašumas** – programa turi turėti grafinę sąsają, kuri būtų pažystama ir nuspėjama mokiniams jau išmokusiems naudotis panašiomis programomis; **funkcionalumas** – turi turėti visas mokykliniam kursui būtinas funkcines galimybes, tame tarpe pažingsninio programų derinimo, reiškinių reikšmių stebėjimo derinimo metu galimybes; **stabilumas** – terpė turi būti išbaigta ir neturėti klaidų; **draugiškumas** – programa daugiausia naudosis neįgudę naudotojai, todėl ji turi turėti aiškią, lengvai pasiekiamą pagalbos sistemą, pastebėti vartotojo daromas klaidas ir apie jas perspėti, padėti išvengti veiksmų galinčių padaryti žalos duomenims; **internacionalizuota** – PĮ turi būti

internacionalizuota tam, kad ją būtų galima lengvai lokalizuoti ir galėtų naudoti kitos šalys [La01][La06].




*22 pav. „FPS“* pagrindinis langas

Tai kad *FPS* programavimo terpė yra pilnai internacionalizuota, leidžia geriau išnaudoti ir kompiliatoriaus internacionalizacijos teikiamus pranašumus. Pavyzdžiui, terpė suteikia galimybę rinkti tekstą ir gauti kompiliatoriaus pranešimus bet kuria kalba.

Naujausioje *FPS* versijoje (0.9.6) naudojamas integruotas *Free Pascal* kompiliatorius, kurio versija 2.1.1. Ši kompiliatoriaus versija dar nėra oficialiai išleista, todėl prie jos tobulinimo galima prisidėti savo pataisomis ir pasiūlymais. Tai yra viena iš priežasčių, kodėl internacionalizavimui buvo pasirinkta ši kompiliatoriaus versija.

## 5.4. *Free Pascal* kompiliatoriaus internacionalizavimas

### 5.4.1. Duomenų kodavimas

Vadovaujantis 3.4 skyriuje pateiktais teiginiais tinkamiausią duomenų koduotę pateikia Unikodo standartas. Unikodo kodavimo metodo diegimą vidinių ir išorinių duomenų kodavimui panagrinėsime atskirai.

**Vidinių duomenų kodavimas**

Pradedant nuo 2.0 versijos *Free Pascal* kompiliatoriuje įdiegtas Unikodo eilučių ir simbolių (*WideString*, *PWideChar*, *WideChar* ir kt.) duomenų tipai, padėjo pamatus tolimesniam kompiliatoriaus suderinamumui su Unikodo standartu diegti. Šie duomenų tipai paremti Unikodo kodavimo metodu UTF-16. Ženklams koduoti juose skiriami 2 baitai. Be to ženklus gali sudaryti ir ženklų sekos, tačiau jų palaikymui būtinos paslaugos (rikiavimo, normalizavimo ir kt.) nerealizuotos. Pastarieji trūkumai nėra aktualūs mokykliniam kursui todėl toliau į juos nebus atsižvelgiama. Įdiegti Unikodo eilučių ir simbolių duomenų tipai nėra pakankamai gerai realizuoti ir bendrąją prasme. Jie turi nemažai trūkumų susijusių su kodavimu, ženklų savybių nustatymu ir kt., kuriuos būtina pašalinti.

Unikodo duomenų tipų palaikymui ir perkodavimui *Free Pascal* naudoja struktūrą *WideStringManager* (18 pav.). Tai įrašo tipo struktūra kurią sudaro funkcinio tipo laukai, skirti operacijoms atlikti – pavyzdžiui, konvertuoti 8 bitų eilutes į Unikodo eilutes, pakeisti raidžių lygį. *Free Pascal* kompiliatorius skirtas įvairioms platformoms ir Unikodo eilučių realizacija priklausoma nuo konkrečios platformos. Todėl *WideStringManager* struktūra skirta atlikti sąsajos vaidmenį tarp įvairių platformų ir kompiliatoriaus. Nagrinėjant jos realizaciją skirtą *Windows* OS pastebėti šie pagrindiniai jos trūkumai:

1. Naudojamos *Windows* API funkcijos nėra perkeliamos į ankstesnes *Windows* OS (jos neveikia *Win9x* sistemose).
2. Nerealizuotas ženklų savybių nustatymas ir eilučių palyginimas. Šios paslaugos yra fundamentalios tolesnei kompiliatoriaus internacionalizacijai.
3. Realizuotos tik raidžių lygio keitimo ir eilučių konvertavimo paslaugos, turi smulkių netikslumų. Pavyzdžiui, laikoma, kad parametrais bus perduodama *PChar* arba *PWideChar* tipo kintamieji, kas nėra teisinga. Parametrais gali būti perduodamos rodyklės į masyvo arba trumpų eilučių tipo duomenis, neturinčius baigiamojo nulinio ženklo.
4. Į struktūrą įtraukta eilė nenaudojamų laukų nesusijusių su Unikodo duomenų tipais.

Paminėti trūkumai buvo pašalinti. Perkeliamumas *Windows* OS atžvilgiu realizuotas panaudojat *Microsoft Layer for Unicode* (apie tai tolimesniame skyrelyje), įdiegtos ženklų savybių nustatymo ir eilučių palyginimo paslaugos, pašalinti realizacijos netikslumai. Pataisyta ir išplėsta *WideStringManager* struktūros realizacija skirta *Windows OS* pateikta faile „unicode.inc“[31].

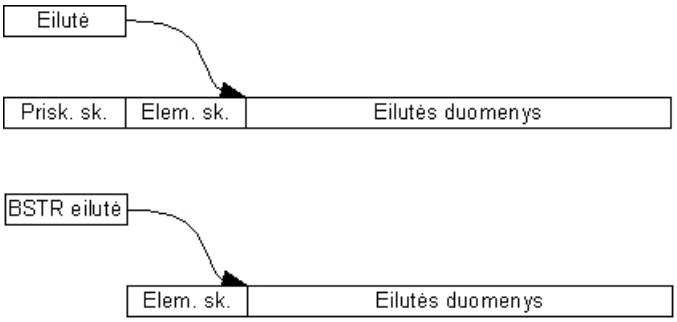
Analizuojant *Free Pascal* kompiliatorių taip pat buvo pastebėta eilė kitų klaidų susijusių su Unikodo duomenų tipais:

1. Klaidingai realizuotas Unikodo eilučių perdavimas parametrų pagalba (testas sudarantis klaidą, pateiktas faile „paramtestas.pas“).
2. Klaidingas konvertavimas, tarp 8 bitų eilučių (simbolių) ir Unikodo eilučių (simbolių).

Realizuojant Unikodo eilutes *Free Pascal* kompiliatoriuje, pavyzdžiu buvo paimtas ilgų eilučių tipas. Tačiau vėliau siekiant suderinamumo su *Windows* OS naudojamu eilučių tipu BSTR (šis suderinamumas svarbus *Free Pascal* kompiliatoriaus suderinamumui su COM technologija palaikyti), jų sandara buvo pakeista. Skirtingai nuo ilgų eilučių tipo (ir pradinės *Free Pascal* Unikodo eilučių versijos) BSTR eilutėse nėra skaičiuojamas jų priskyrimo skaičius (20 pav.). Šių eilučių sukūrimui naudojamos OS paslaugos ir iš dalies pati OS rūpinasi jų panaikinimu iš atminties. Dėl to autoriams neapsižiūrėjus atsirado Unikodo eilučių perdavimo parametrų pagalba klaida, nes vis dar buvo laikoma, kad yra skaičiuojamas eilučių priskyrimo skaičius. Šios klaidos pataisa pateikta faile „param.diff“.

Klaidingo konvertavimo klaidos atsiranda priskiriant eilučių arba simbolių kintamiesiems (parametrams) skirtingo tipo konstantas arba kintamuosius. Faile „konvtestas.pas“ pateiktas testas perrenka visus galimus priskyrimo variantus. Esant konvertavimo klaidoms išvedami duomenys skirtųsi nuo priskirtų. Testuojant *Free Pascal* kompiliatorių paaiškėjo, kad jis klaidingai priskiria simbolių konstantas, 8 bitų simbolių tipo kintamiesiems ir klaidingai konvertuoja 8-bitų eilutes (ir simbolius) į Unikodo eilutes (ir simbolius) ir atvirkščiai.

[31] Pataisų failai pateikti prie disertacijos pridėto CD pakatalogyje „Pataisos“.




*23 pav*. Eilučių realizacija

Taisant šias klaidas pasinaudota jau prieš tai pataisytos *WideStringManager* struktūros teikiamomis paslaugomis, įdiegiant konvertavimui skirtas funkcijas vykdymo meto bibliotekoje. Taip pat buvo pataisytas bei papildytas transliatoriaus semantinis analizatorius tam, kad šios funkcijos būtų įkompiliuojamos į išvedinius. Vykdymo meto bibliotekos pataisa pateikta faile „konvvmb.diff“, transliatoriaus pataisa pateikta faile „konv.diff“.

**Išorinių duomenų kodavimas**

Pirminio teksto kodavimui *Free Pascal* kompiliatorius naudoja 8 bitų arba UTF-8 metodus (koduotė nurodoma kompiliatoriaus parinkčių pagalba). UTF-8 Unikodo kodavimo metodas pasirinktas todėl, kad nereikalauja pilnai perrašyti leksikos analizatorių arba perkoduoti pirminį tekstą (atliekamas tik eilučių konstantų perkodavimas į UTF-16). Pakanka išplėsti jau turimą 8 bitų analizatorių, nes naudojant UTF-8 metodą ženklai taip pat koduojami 8 bitų ženklais arba jų sekomis. Be to tokiu atveju išplėtimus lengviau įdiegti, nes nebūtina juos įdiegti visus iš karto. Tai galima atlikti palaipsniui.

Diegiant Unikodo kodavimą numatyta atlikti šiuos leksikos analizatoriaus išplėtimus:

1. Įdiegti Unikodo standartą atitinkančią vardų sintaksę.
2. Įdiegti Unikodo standartą atitinkančius matematinių operacijų sinonimus.

*Free Pascal* kompiliatoriuje pirminio teksto leksikos analizę atlieka klasė *TScannerFile*. Internacionalizuojant kompiliatorių buvo sukurta jos klasė paveldėtoja *TUnicodeScannerFile*, kuri nustelbia dalį protėvio metodų, metodais atliekančiais Unikodu koduoto teksto leksikos analizę. Kompiliatoriaus inicializacijos metu nustatoma pirminio teksto koduotė ir sukuriamas atitinkamos klasės leksikos analizatoriaus egzempliorius.

Unikodo standarte apibrėžtos vardų sintaksės taisyklės suteikia tam tikrą jų realizacijos laisvę atsižvelgiant į programavimo kalbą bei suderinamumo su Unikodu lygį (paaiškinta 4.5 skyrelyje). Į tai atsižvelgiant *Free Pascal* įdiegtos šiek tiek griežčiau apibrėžtos sintaksės taisyklės priderintos prie Paskalio kalbos ir neleidžiančios varduose naudoti „nesąmoningų ženklų“:

```
<vardas> ::= <pirmas vardo simbolis>(<tolesnis vardo simbolis>)*
<pirmas vardo simbolis> ::= „_“ | <raidė>
<tolesnis vardo simbolis> ::= „_“ | <raidė> | <skaitmuo>
```

Pažymėtina, kad šiose taisyklėse apibrėžtas raidės simbolis gali reikšti bet kurio rašto raides (tame tarpe ir hieroglifus), o skaitmens simbolis bet kurio rašto skaitmenį.

Diegiant matematinių operacijų sinonimus įdiegti atitikmenys pateikti sekančioje lentelėje (testas skirtas jiems patikrinti pateiktas faile „oper.pas“:

*11 lentelė*. Matematinių operacijų sinonimų pavyzdžiai

| Pascal žymenys | Matematiniai žymenys | Matematinių žymenų kodai Unikode |
|---|---|---|
| := | ← | U+2190 |
| In | ∈ | U+2208 |
| Div | ÷ | U+00F7 |
| * | × | U+00D7 |
| Not | ¬ | U+00AC |
| <= | ≤ | U+2264 |
| >= | ≥ | U+2265 |
| <> | ≠ | U+2260 |
| And | ∧ | U+2227 |
| Or | ∨ | U+2228 |
| Xor | ⊻ | U+22BB |
| ^ | ↑ | U+2191 |

Atlikti kompiliatoriaus išplėtimai pateikti faile „scan.diff“.

Išoriniais duomenimis taip pat galima laikyti ir *Free Pascal* kompiliatoriaus konfigūracinius ir žurnalų failus. Konfigūraciniai failai sukuriami naudotojo juose nurodant kompiliatoriaus parinktis. Kompiliatoriaus parinktis sudarančiais komponentais gali būti failų vardai, todėl jų, kaip ir žurnalų failų kodavimui, kuriuose pateikiama kompiliatoriaus veiklos ataskaita, turi būti naudojama Unikodo koduotė. Šis išplėtimas įdiegtas internacionalizuojant įvedimo ir išvedimo paslaugą (žr. tolesnį skyrelį).

### 5.4.2. Įvedimas ir išvedimas

Įvedimui ir išvedimui tiek į failą, tiek į ekraną naudojamos tos pačios paslaugos savo ruožtu naudojančios OS paslaugas, todėl jų realizacija kiekvienos OS atžvilgiu skiriasi. Panagrinėsime *Windows OS* skirtų paslaugų realizaciją.

*Free Pascal* vykdymo meto biblioteka turi dvi pagrindines funkcijas skirtas įvedimui ir išvedimui tai *read* ir *write*. Jos apibrėžtos kaip kompiliatoriaus simboliai, o įvedimo arba išvedimo metu kompiliatorius naudoja įvedamų arba išvedamų duomenų tipą atitinkančias jų realizacijas esančias vykdymo meto bibliotekoje. Informacijai (tarpiniam duomenų saugojimui, failų pavadinimams ir identifikatoriams, eilučių pabaigoms) saugoti šios funkcijos naudoja struktūras *TextRec* bei *FileRec* (išvedant netekstinius duomenis).

Paslauga taip pat pateikia keletą funkcijų skirtų atlikti veiksmus su failais: susieti (*Assign*), pervadinti (*Rename*), kuriose naudojami failų vardai todėl jas taip pat būtina išplėsti.

Analizuojant paslaugos realizaciją pastebėta svarbių jos trūkumų:

1) Išvedami duomenys koduojami tik 8 bitais (net jei tai Unikodo eilutės arba simboliai).
2) Nerealizuotas Unikodo eilučių ir simbolių įvedimas.
3) Failų vardai koduojami 8 bitais.

Šalinant šiuos trūkumus buvo atlikti šie išplėtimai:

1) Išplėstos *TextRec* ir *FileRec* struktūros.

*TextRec* struktūra išplėsta pakeičiant laikinai joje saugomo teksto, failų vardų ir eilučių pabaigų kodavimo būdą į UTF-16. *FileRec* struktūra skirta netekstiniams failams. Ji išplėsta pakeičiant failų vardų kodavimą. Pataisos pateiktos faile „textrec.diff“.

Kompiliatoriaus nustato duomenų tipų suderinamumą su minėtomis struktūromis pagal jų užimamos atminties kiekį, todėl jas pakeitus būtina kompiliatoriui nurodyti naują jų dydį. Pataisa „symdef.diff“.

2) Išplėstos ir suderintos su Unikodu įvedimo ir išvedimo paslaugos funkcijos.

*Read* ir *Write* funkcijų pradinė realizacija yra nepriklausomos nuo platformos. Duomenys nėra tiesiogiai įvedami arba išvedami naudojant OS paslaugas. Iš pradžių jie laikinai išsaugomi *TextRec* struktūroje ir tik vėliau galutinai apdorojami. Tokiu būdu sąsajos su OS paslaugomis vaidmenį atlieka antrinės funkcijos kurios bus aprašytos toliau.

Pakeitus *TextRec* struktūros laikinos duomenų saugyklos tipą pakito ir į jį įrašomų duomenų kodavimas, todėl daugelyje vietų teko jį pakeisti panaudojant *WideStringManager* struktūros paslaugas. Taip pat buvo pakeistas eilučių pabaigų nustatymas.

Papildomai įdiegta galimybė nurodyti į failą išvedamų arba iš jo skaitomų duomenų koduotę. Ši galimybė leidžia pasirinkti vieną iš trijų duomenų kodavimo tipų: UTF-16, UTF-8 ir 8 bitais. Suderinamumui su ankstesnėmis programomis palaikyti pagal nutylėjimą naudojamas 8 bitų kodavimas, tačiau taip pat buvo numatyta galimybė nustatyti kitą numatytąjį kodavimą.

Be jau minėtų išplėtimų pakeisti failų vardų tipai bei įdiegtos galimybės įvesti Unikodo eilutes ir simbolius.

Visus šiuos išplėtimus realizuojanti pataisa pateikta faile „text.diff“.

3) Išplėsta *Windows* OS skirta įvedimo ir išvedimo paslaugos realizacija.

Išskirsime dvi paslaugos realizacijas, tai 1) įvedimas ir išvedimas į failą ir 2) į ekraną (konsolę).

Įvedimo ir išvedimo į failą palaugos realizacija išplėsta įdiegiant galimybę pasirinkti įvedamo arba išvedamo teksto kodavimo būdą. Taip pat pakeisti failų vardų tipai.

Įvedimo ir išvedimo į ekraną paslaugos realizaciją taip pat galima išskirti į dvi dalis:

1) Bazinė realizuota *System* modulyje
2) Išplėstinė realizuota *Crt* modulyje.

Pagrindinis jų skirtumas tas, kad bazinė realizacija naudoja aukštesnio lygio OS paslaugas – skirta tik įvesti ir išvesti tekstą. Tuo tarpu išplėstinė naudoja žemesnio lygio paslaugas ir leidžia atlikti daugiau veiksmų, pavyzdžiui nurodyti išvedamo teksto vietą ekrane, jo spalvą.

Abi realizacijos išplėstos pakeičiant iki tol naudotas OS palaugas į paslaugas suderintas su Unikodu.

Išplėtimus realizuojančios pataisos pateiktos failuose „iosystem.diff“ ir „iocrt.diff“.

### 5.4.3. Leksikos elementai

Vienas iš pagrindinių uždavinių internacionalizuojant leksikos elementus yra sudaryti galimybes bazinių žodžių lokalizavimui. *Free Pascal* baziniai žodžiai užkoduoti pirminiame tekste, masyve. Be to jų internacionalizavimą apsunkina specifinė šio masyvo sandara. Pakeitus žodžių tvarką masyve, jie turi būti iš naujo suindeksuojami tam, kad kompiliatorius leksinės analizės metu galėtų sparčiau vykdyti jų paiešką.

Internacionalizavimo metu baziniai žodžiai buvo atskirti nuo pirminio teksto ir iškelti į išorinį failą, koduotą XML formatu. Informacija iš šio failo nuskaitoma kompiliatoriaus vykdymo metu. Kurios lokalės leksiką įkelti nurodoma naujai įdiegtos kompiliatoriaus parinkties (-L<leksikos lokalės identifikatorius>) pagalba. XML formatas pasirinktas dėl jo atvirumo ir paprastumo. Šiuo formatu koduotus failus galima lokalizuoti pasitelkiant standartines priemones. Įdiegtos leksikos lokalės taip pat registruojamos XML formatu koduotame faile.

Šiuos išplėtimus realizuojanti pataisa bei leksikos lokalės ir jų registracijos failų pavyzdžiai pateikti failuose „tokens.diff“, „tockensinfo.inc“, „en.xml“, „locales.pas“ ir „locales.xml“.

Kompiliatoriaus leksikos lokalės informaciją sudaro ne tik baziniai žodžiai, bet ir formatų elementai. Šio etapo metu įdiegtas pagrindinis (dažniausiai naudojamas) formato elementas nurodantis dešimtainės trupmenos skyriklio ženklą. Jis įdiegimas pareikalavo leksikos analizatoriaus išplėtimų. Šie išplėtimai pateikti faile „separator.diff“.

### 5.4.4. Lokalės elementai

*Free Pascal* kompiliatorius naudoja gana daug lokalės elementų, tačiau svarbiausias ir dažniausiai naudojamas iš jų yra dešimtainės trupmenos skyriklis. Jį internacionalizuojant panaudota *Windows* OS lokalės informacija, o dešimtainės trupmenos skyriklis automatiškai parenkamas atsižvelgiant į vartotojo nustatytą lokalę. Tokiu būdu sudaryta galimybė teisingai lokalės atžvilgiu įvesti ir išvesti dešimtainius skaičius. Šie išplėtimai pateikti faile „decsep.diff“.

### 5.4.5. Išteklių atskyrimas

Pagrindinius *Free Pascal* kompiliatoriaus išteklius sudaro pranešimų tekstas. Jie atskirti nuo pirminio teksto į išorinį specifinio formato *(.msg*) failą ir naudojamas specifinis jų įkėlimo mechanizmas. Pastebėta keletas esminių trūkumų:

1) Dėl specifinio formato pranešimų failą sunku lokalizuoti, tam negali būti taikomos standartinės priemonės.
2) Dėl specifinio pranešimų įkėlimo mechanizmo naujai pridedamus kompiliatoriaus pranešimus sunku atskirti nuo pirminio kompiliatoriaus teksto.
3) Pranešimai koduojami 8 bitais.
4) Ne visi kompiliatoriaus pranešimai atskirti nuo pirminio teksto.

Šalinant šiuos trūkumus atlikti šie pakeitimai:

1) Pranešimų failai konvertuoti į *GetText* tipo išteklių failus.
2) Įdiegtas *GetText* pranešimų įkėlimo mechanizmas.
3) Pakeistas pranešimų kodavimas į Unikodą.
4) Pagrindiniai lokalizuotini pranešimai atskirti nuo pirminio teksto.

Naujas pranešimų formatas ir įkėlimo mechanizmas yra žymiai pranašesni už naudotus anksčiau. Išteklių failus galima lengvai lokalizuoti pasitelkiant standartines priemones, pavyzdžiui *poEdit*. Nesudėtingu tapo ir pranešimų atskyrimas papildant kompiliatorių naujais

pranešimais. Naujus pranešimus tereikia aprašyti *resourcestring* srityje ir įtraukti į *GetText* pranešimų katalogo šablono failą. Pažymėtina, kad *Free Pascal* naudojamas *GetText* pranešimų įkėlimo mechanizmas neturi nevienareikšmių pranešimų trūkumo (žr. 4.6 skyrelį), nes įkeliant pranešimus yra atsižvelgiama į *resourcestring* srityje jiems priskirtus identifikatorius, kuriems Paskalio kalbos sintaksė neleidžia kartotis.

Įdiegus *GetText* metodą, kaip ir anksčiau, konkrečios lokalės pranešimų failą galima nurodyti naudojant kompiliatoriaus parinktis. Jei naudojant parinktis failas nenurodomas, tuomet, skirtingai nei anksčiau, kompiliatorius automatiškai parenka ir įkelia OS nustatytos vartotojo lokalės pranešimų failą.

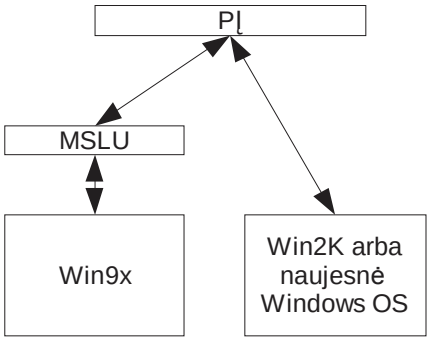
Išplėtimai atlikti diegiant *GetText* išteklių atskyrimo metodą ir pakeičiant kodavimą pateikti failuose „verbose.diff“, „msgs.pas“.

### 5.4.6. Platformos paslaugos

*Free Pascal* kompiliatoriaus naudoja įvedimo ir išvedimo, failų sistemos, kodavimo, lokalės ir kt. platformos paslaugas. Dauguma platformų pateikia pakankamai internacionalizuotas paslaugas, tačiau ir tokiu atveju jos gali būti klaidingai naudojamos. Pavyzdžiui, analizuojant *Windows* OS skirtą *Free Pascal* kompiliatorių pastebėti jo trūkumai:

1. Daugiausia naudojama 8 bitų *Windows* API.
2. Naudojama Unikodinė *Windows* API nėra perkeliama *Windows* OS atžvilgiu (neveikia *Win9x* OS).

Šalinant šiuos trūkumus pagrindinės 8 bitų *Windows* API funkcijos pakeistos į unikodines. Perkeliamumui *Windows* OS atžvilgiu realizuoti įdiegta *Microsoft Layer for Unicode* (MSLU) biblioteka. Ji naudojama tik *Win9x* OS, kurios nepalaiko Unikodo.




*24 pav*. Operacinės sistemos paslaugų naudojimas

Išplėtimai atlikti šalinant šiuos trūkumus pateikti failuose „system.diff“ ir „unikode.inc“.

## 5.5. Išvados

1. Sukauptas žinias pavyko sėkmingai pritaikyti internacionalizuojant *Free Pascal* iki užsibrėžto lygio.
2. Buvo internacionalizuotas ne tik *Free Pascal* kompiliatorius, bet ir jam skirta programavimo terpė *FPS*, tokiu būdu ji suteikia galimybę išnaudoti kompiliatoriaus internacionalizacijos teikiamus pranašumus, todėl jie gali būti dar sėkmingiau taikomi mokymui.
3. Nors atlikti darbai nėra didelės apimties lyginant su pilnu kompiliatoriaus internacionalizavimu, tačiau ir jie pareikalavo gana daug išplėtimų. Išplėtimų teksto eilučių skaičius siekia keletą tūkstančių.

4. Bendravimas su kompiliatoriaus autoriais atskleidė jų atsainų požiūrį į kompiliatoriaus internacionalizaciją. Paaiškėjo, kad autoriai labiau linkę teikti pirmenybę kompiliatoriaus efektyvumui, nei praktiškumui, net ir tais atvejais kai dėl pataisų susijusių su internacionalizacija (pavyzdžiui, Unikodu koduoto teksto įvedimo ir išvedimo) efektyvumas nukenčia visai nedaug, o išaugęs praktiškumas duotų naudos daugumai naudotojų.

# BENDROSIOS IŠVADOS IR REZULTATAI

1. Sukurtas ir pateiktas kompiliatorių internacionalizuotumo lygio vertinimo metodas. Juo remiantis atliktas kompiliatorių internacionalizuotumo tyrimas atskleidė itin žemą jų internacionalizacijos lygį. Ištirti kompiliatoriai pilnai tenkino tik 10 %, nepakankamai tenkino 58 % ir visai netenkino 32 % reikalavimų. Nustatyta, kad kompiliatorių internacionalizuotumo lygis įtakoja jais kuriamos programinės įrangos internacionalizuotumo lygį. Dėl to kuriant šiais kompiliatoriais programinę įrangą jos internacionalizavimui būtinos papildomos sąnaudos. Remiantis šaltiniais jos gali sudaryti 10–20 % programinės įrangos gamybos sąnaudų.
2. Nustatyta, kad didelę dalį programų internacionalizavimo elementų galima perkelti į kompiliatorių internacionalizavimą ir taip padidinti jais gaminamos programinės įrangos internacionalizuotumo lygį bei taip sumažinti jos internacionalizavimui reikalingas sąnaudas.
3. Sukurtas ir pateiktas siūlomas kompiliatorių internacionalizavimo metodas, kuris yra paremtas susistemintomis, naujausiomis kompiliatorių internacionalizavimo srities žiniomis, sukauptomis teorinės analizės, atliktų kompiliatorių internacionalizuotumo lygio tyrimo ir eksperimentinio kompiliatoriaus internacionalizavimo metu.
4. Eksperimentiškai internacionalizuotas kompiliatorius *Free Pascal*. Šis kompiliatorius naudojamas mokymui Lietuvos mokyklose, o internacionalizacija suteikia jam pranašumų. Sukurta grafinė internacionalizuota programavimo terpė į kurią integruotas internacionalizuotas *Free Pascal* kompiliatorius. Ji leidžia geriau išnaudoti kompiliatoriaus internacionalizacijos teikiamus pranašumus ir sėkmingiau taikyti jį mokymui.

# LITERATŪROS SĄRAŠAS


[AG00] Aaron M. ir Gould E. W. (2000). *Crosscurrents: cultural dimensions and global Web user-interface design*. ACM Interactions 7(4), p. 32–46.

[Ag02] Agejevas A. (2002*). Lietuvių kalbos rašybos tikrinimas atviro kodo priemonėmis*. http://www.maf.vu.lt/~alga/lt-ispell.pdf

[Ap06] Apple (2006). *Technical Note OV20: Internationalization Checklist*. http://developer.apple.com/technotes/ov/pdf/ov_20.pdf

[AS01] Atkin S. ir Stansifer R. (2001). *Implementations of bidirectional reordering algorithms*. In Unicode Consortium (Ed.), Eighteenth International Unicode Conference (IUC18) Unicode and the Web: the Global Connection, Hong Kong. San Jose, California, The Unicode Consortium.

[BB98] Barber W, ir Badre A. (1998). *Culturability: The Merging of Culture and Usability*. Proceedings of the 4th Conference on Human Factors and the Web, Basking Ridge.

[Be90] Becker J. D. (1990). *Arabic Word Processing*. Communzcatzons of the Assoczatzon for Computzng Machanery, 30(7), p. 600–610.

[Bo03] Bouckaert R. (2003). *A Probabilistic Line Breaking Algorithm*. In D. Gedeon, L. Fung. (Eds.), Lecture Notes in Computer Science. AI 2003: Advances in Artificial Intelligence: 16th Australian Conference on AI, Perth, Australia, p. 390-401.

[Br96] Bright W. (1996). *The Devanagari script*. In P. Daniels, W. Bright (Eds.), The World's Writing Systems. Oxford University Press, New York, NY, p. 384–390.

[CS98] Choong Y.-Y. ir Salvendy G. (1998). *Design of Icons for use by Chinese in mainland China*. Interacting with Computers, Special Issue: Shared values and shared interfaces, 9, 4, p. 417–430.

[DJG05] Dagienė V., Jevsikova T. ir Grigas G. (2005). *Enciklopedinis kompiuterijos žodynas*. Vilnius: TEV, 104 p.

[DL01] Dagienė V. ir Laucius R. (2001). *„Free Pascal" panaudojimas informatikos kursui*. Lietuvos matematikos rinkinys, 41 t., spec. nr., p. 267–271.

[DL04] Dagienė V. ir Laucius R. (2004). *Internationalization of open source software: framework and some issues.* In: T. Boyle, P. Oriogun, A. Pakstas (Eds.), 2nd International Conference Information Technology: Research and Education, London, London Metropolitan University, p. 204–207.

[DB96] Daniels P. T. ir Bright W. (1996). *The World's Writing Systems*. Oxford University Press, 968 p.

[Da05] David N. W. (2005). *Programming Language Popularity*. http://www.dedasys.com/articles/language_popularity.html

[DR01] Dershowitz N. ir Reingold E. M. (2001). *Calendrical Calculations*. Cambridge University Press, 423 p.

[DVW03] Dybdahl L. B., Vasquez J. G., Wendt S. (2003). GNU *gettext for Delphi, C++ Builder and Kylix*. http://dybdahl.dk/dxgettext/docs/manual.pdf

[Dr03] Dr. International (2003). *Developing International Software*. Second Edition, Microsoft Press, 1040 p.

[DMP02] Drepper U., Meyering J., Pinard F., Haible B. (2002). *GNUgetext tools*. http://www.gnu.org/software/gettext/manual/ps/gettext.ps.gz

[Es02] Esselink B. (2002*). Localization and translation.* Computers and translation. John Benjamins Publishing Company, p. 67–87.

[Es03] Esselink B. (2003). *The evolution of localization*. Multilingual computing and technology 57 (supplement), p. 4–7.

[Ev98] Evers V. (1998). *Cross-cultural understanding of metaphors in interface design*. In Ess, C. and Sudweeks, F. (Eds.), Proceedings CATAC'98, Cultural Attitudes towards

Technology and Communication, science Museum, London. University of Sydney, Australia, p. 261-262.
[Ev01] Evers V. (2001). *Cultural aspects of user interface understanding*. Doctoral Dissertation. Open University, London, England, 377 p.
[ED97] Evers V. ir Day D. (1997). *The role of culture in interface acceptance*. In Howard, S., Hammond, J., and Lindgaard, G. (Eds.), Human-Computer Interaction: Interact '97, London, Chapman & Hall, p. 260-267.
[Fi00] Fine J. (2000*). Line breaking and page breaking*. TUGboat, 21(3), p. 210–221.
[FG03] Ford G. ir Gelderblom J.H. (2003). *The effects of culture on performance Achieved through the use ofhuman computer interaction*, proceedings of SAICSIT, p. 218–230
[Go05] Gordon R. G. (2005). *Ethnologue: Languages of the World*. 15th edition, SIL International, 1272 p.
[GL85] Gould J. D. ir Lewis C. (1985). *Designing for usability: Key principles and what designers think*. Communications of the ACM, 28 (3), p. 300–311.
[Gr93] Greenwood T. G. (1993). *International cultural differences in software.* Digital Technical Journal 5(3), p. 8–20.
[Gr97] Gribbons W. (1997*). Designing for the Global Community*. Proceedings. of IEEE Intl. Professional Communication Conference, (Salt Lake City, US), p. 261–273.
[Gr03] Grigas G. (2003) *Interneto programų paketo lietuvinimo patirtis*. Informacinės technologijos: Konferencijos pranešimų medžiaga. Kaunas: KTU, p. I-15 – I-21
[Gr00] Grigas G.(2000). *Programavimo kalbų leksikos elementų analizė mokymo požiūriu*. Informatika 1(35), p. 45–67.
[Ha59] Hall E. (1959) *The Silent Language*, Doubleday, New York, 224 p.
[HH97] Hall P. V. ir Hudson R. (1997*). Software without frontiers: A Multi-Platform, Multi-Cultural, Multi-Nation Approach*. Willey & Sons, 350 p.
[Ha02] Hansmeyer C. (2002). *The History of Phonetic Alphabets*. University of Bremen, 15 p.
[Ho80] Hofstede G. (1980). *Culture's Consequences: International Differences in Work-Related Values*. Sage, Beverly Hills, California, 328 p.
[Ho96] Hofstede G. (1996). *Cultures and organisations: software of the mind*. New York: McGraw Hill, 279 p.
[Ho96a] Hoft N. (1996). *Developing a Cultural Model. International User Interfaces*, edited by del Galdo E.M. & Nielsen J., John Wiley & Sons, p. 41–73.
[HJP03] Hogan J.M., Ho-Stuart C. and Pham B. (2003 *). Current Issues in Software Internationalisation*. The Australian Computer Science Conference, Adelaide.
[IBM05] IBM and Others (2005). *ICU User Guide*. http://icu.sourceforge.net/userguide/ icu.pdf
[IS03] Immonen J. ir Sajaniemi J. (2003). *Software Globalisation in Finland: A State-of-the-practice Survey*. University of Joensuu, Department of Computer Science, Technical Report, Series A, Report A-2003-1.
[IPA99] IPA[1] (1999). *The Handbook of the International Phonetic Association*. Cambridge University Press, 214 p.
[IN96] Ito M. ir Nakakoji K. (1996). *Impact of culture on user interface design*. In del Galdo, E. M. and Nielson, J. (Eds.), International User Interfaces, John Wiley & Sons, New York, p. 105–126.
[Ye96] Yeo A. (1996). *World-wide CHI: Cultural user interfaces, a silver lining in cultural diversity*. SIGCHI, 28(3), p. 4–7.
[Ye01] Yeo A. (2001). *Global-Software Development Lifecycle: An Exploratory Study*. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, p. 104–111.

[Yo01] Young E. (2001). *A Framework for the Integration of Internationalization into the Software Development Process*. Disertacija, Pietų Dakotos Universitetas, 196 p.
[JSD04] Jagne J., Smith S. G., Duncker E. ir Curzon P. (2004). *Cross-cultural interface design strategy*. (IDC-TR-2004-006): Middlesex University, 2004.
[Ka95] Kano, N. (1995). *Developing International Software for Windows 95 and Windows NT*. Microsoft Press, Redmond, Washington, 300 p.
[Ka01] Käpyaho J., (2001). *Internationalisation in Operating Systems for Handheld Devices*. Master thesis, University of Tampere, Department of Computer and Information Sciences, 80 p.
[KW01] Kaplan M. ir Wissink C. (2001*). MSLU: Develop Unicode Applications for Windows 9x Platforms with the Microsoft Layer for Unicode*. MSDN Magazine. Vol. 16. Nr. 10.
[Ke98] Keniston K. (1998). Cultural Diversity or Global Monoculture: The Impacts of the Information Age. The Global Village Conference. Bangalore, India.
[Ke98a] Keniston K. (1998). *Politics, culture and software*. Economic and Political Weekly, (January 17).
[Ka99] Keniston K. (1999). *Language, Power and Software*. Cultural Attitudes Towards Technology and Communication. New York: Suny Press.
[KKR02] Kersten G. E., Kersten M. A. and Rakowski W. M. (2002). *Application Software and Culture: Beyond the Surface of Software Interface*. Journal of Global Information Management, InterNeg Reports INR1/01.
[KMN00] Kersten G.E., S. Matwin S. Noronha ir Kersten M.A. (2000). *The Software for Cultures and the Cultures in Software*. Proceedings of the 8th European Conference on Information System. ECIS2000, H.R. Hansen, M. Bichler and H. Harald, (Eds.). Vienna University of Economics and Business Administration. Vol. 1, p. 509–514.
[Kl62] Kluckhohn C. (1962). *Culture and Behaviour*. University of Arizona Press, Tucson.
[KG97] Knight K. ir Graehl J. (1997). Machine transliteration. Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, Madrid, Spain, p.128–135.
[KK52] Kroeber A. L. ir Kluckhohn C. (1952). *Culture: A Critical Review of Concepts and Definitions*. Peabody Museum, Cambridge, Massachusetts, United States.
[KP58] Kroeber A. L. ir Parsons T. (1958). *The concepts of culture and of social systems*. American Sociological Review, 23, p. 582.
[La01] Laucius R. (2001). *Kompiliatoriaus Free Pascal sąsajos su vartotoju*. Kompiuterininkų dienos – 2001, p. 113–115.
[La03] Laucius R. (2003). *Hiperteksto rašyklių palyginimas lokalizavimo galimybių požiūriu.* Lietuvos matem. rink. 43 (spec. nr.), p. 254–25
[La03] Laucius R. (2003). *Lokalės, jų sandara ir ypatumai*. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, Kaunas: Technologija, p. I-(1–7).
[LD03] Laucius R. ir Dagienė V. (2003). *Raštinės programinės įrangos „OpenOffice.org" adaptavimas lokalės normoms*. Informacijos mokslai, 26, p. 240–245.
[LD05] Laucius R., Dagienė V. (2005). *Lokalizavimo kurso projektavimas.* Lietuvos matematikos rinkinys, 45, spec. nr., p. 213–218
[La06] Laucius R. (2006). *Issues of Selecting a Programming Environment for a Programming Curriculum in General Education*. In: R. T. Mittermeir (Ed.), *Lect. Notes in Computer Science*, vol. 4226. Informatics Education - The Bridge between Using and Understanding Computers, p. 169–178.
[LISA06] LISA (2006). *Frequently Asked Questions about LISA and the Localization Industry*. http://www.lisa.org/info/faqs.html

[LS01] Loeschky D. ir Senthil S. (2001). *Universal I18n Framework for Office Applications*. Sun Microsystems, Inc., 2001.
[Lo94] Lovgren J. (1994). *How to choose good metaphors*. IEEE Software, 11(3), p. 86–88.
[MJ98] Mahemoff M. J. ir Johnston L. J. (1998). *Software Internationalisation: Implications for Requirements Engineering*. Proceedings of the Third Australian Workshop on Requirements Engineering, Deakin University: Geelong, p. 83–90.
[Ma96] Marcus A. (1996). *Icon and symbol design issues for graphical user interfaces*. In del Galdo, E. M. and Nielson, J. (Eds.), International User Interfaces, John Wiley & Sons, New York, p. 257–270..
[Mi99] Microsoft (1999*). Microsoft Portable Executable and Common Object File Format Specification*. Microsoft Corporation, Revision 8.0 – May 16, 2006, 65 p.
[Mi03] Microsoft (2003*). October 2003 MSDN Library. Microsoft Corporation*. DVD.
[Mi04] Microsoft (2004). *Microsoft Typography – Specifications: Overview*. http://www.microsoft.com/typography/SpecificationsOverview.mspx
[Mi06] Microsoft (2006). *Globalization Step-by-Step: Testing for World-Readiness Overview*. Microsoft, Global Development and Computing Portal.
[Mi06a] Microsoft (2006). *Resources in .Resx File Format. .NET Framework* Developer's Guide.
[Mi96] Mitton R. (1996). *Spellchecking by computer*. J. Simplif. Spell. Soc., Vol. 20. No. 1, p. 4–11.
[MM85] Myers I. B., ir McCaulley M. H. (1985). *A guide to the development and use of the Myers-Briggs Type Indicator*. Consulting Psychologist Press, Palo Alto, CA, 1985.
[Su01] O'Sullivan P. (2001). A Paradigm for Creating Multilingual Interfaces. Thesis submited to The University of Limerick.
[OASIS03] OASIS (2003). *XLIFF 1.1 Specification*.
[Co98] O'Conner J. (1998). *Java Internationalization: Localization with ResourceBundles*. http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles/
[RY03] Rätzmann M. ir Young C. (2003). *Software Testing and Internationalization. Lemoine International and the Localization Industry Standards Association (LISA)*. http://www.lisa.org/interact/2003/SoftwareTesting.zip;
[RW01] Rowe G. and Wright G. (2001). *Expert opinions in forecasting: The role of the Delphi technique*. J. S. Armstrong (ed.), Principles of Forecasting. Boston: Kluwer Academic Publishers, p. 125–144.
[Ro00] Roach P. (2000). *English Phonetics and Phonology*. Cambridge University Press, Cambridge, 192 p.
[Ro03] Rolfe R. (2003). *What is an IME (Input Method Editor) and how do I use it?* http://www.microsoft.com/globaldev/handson/user/IME_Paper.mspx
[Ru06] Rubic (2006).*Localization-readiness testing checklist*. http://www.rubric.com/en/pdfs/Rubric_local_readiness.pdf
[RB93] Russo P. ir Boor S. (1993). *How Fluent is Your Interface? Designing for International Users*. Proceedings of INTERCHI´93, ACM Press, p. 342–347.
[SHH04] Sahama T., Ho-Stuart C. and Hogan J. M. (2004). *Developing and delivering a software internationalisation subject*. Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32. Dunedin, New Zealand.
[Sa49] Sapir E. (1949) *Selected Writings of Edward Sapir*. University of California Press, Berkeley.
[Sc02] Schäler R. (2002). *The Cultural Dimension in Software Localisation*. Localisation Focus, v. 1, issue 2

[SY00] Schirato T. ir Yell S. (2000). *Communication and Culture: An introduction*. Sage Publications, London.
[Sc00] Schmitt D. A. (2000). *International Programming for Microsoft Windows*. Microsoft Press.
[Sh00] Shen J. (2000). *User Interface Internationalization*. CIS732 Final Project, 2000.
[SC84] Sitton S., ir Chmelir G. (1984). *The intuitive computer programmer*. Datamation, 15.
[Sp03] Sproat R. (2003). *A Formal Computational Analysis of Indic Scripts*. International Symposium on Indic Scripts: Past and Future, Tokyo.
[St04] Stephan D. (2004). *Intercultural Research: The Current State of Knowledge*. London: Middlesex University.
[Su02] Sun (2001). *Software Product Internationalization Taxonomy*. http://developers.sun.com/dev/gadc/des_dev/i18ntaxonomy/i18n_taxonomy.pdf
[Su06] Sun (2006). *Internationalization Verification Checklists*. http://developers.sun.com/dev/gadc/i18ntesting/checklists/
[Ta93] Taylor D. (1993). *Global Software: Developing Applications for the International Market*. Springer-Verlag.
[Th01] Thomas J. R. (2001*). Technical Overview of the Kylix Compiler*.
[TS05] Thorkil C., Shivam G. (2005). *Cross cultural usability testing - the relationship between evaluator and test user*. Working paper, 2005.
[TIOBE06] TIOBE (2006*). TIOBE Programming Community Index for August 2006*. http://www.tiobe.com/index.htm?tiobe_index
[TIS95] TIS (1995). *ELF: Executable and Linkable Format: Portable Formats Specification*. Version 1.2. Tools Interface Standards (TIS) Committee.
[To92] Tognazzini, B. (1992). *Tog on Interface*. Addison-Wesley, Reading, MA, 1992.
[Tr72] Triandis H.C. (Red.) (1972). *The Analysis of Subjective Culture*. Wiley, New York.
[Tr99] Trillo N.G. (1999). *The Cultural Component of Designing and Evaluating International User Interfaces*. Proceedings of the 32nd Hawaii International Conference on System Sciences, Hawaii.
[Tr93] Trompenaars F. (1993). *Riding the Waves of Culture*. Nicholas Brealey Publishing, London.
[TDD95] Tuoc V., David J.T. ir Driscoll K.(1995). *Internationalization: Developing Software for Global Markets*. John Wiley & Sons, Inc..
[Un03] Unicode (2003). *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 1504 p.
[Va02] Vatrapu R. (2002). *Culture and International Usability Testing: The effects of Culture in Structured Interviews*. Master thesis,Virginia Polytechnic Institute and State University.
[VV77] Voegelin C. F. ir Voegelin F. M. (1977). *Classification and index of the world's languages*. New York: Elsevier.
[Vo98] Vohringer-Kuhnt T. (1998). *The Influence of Culture on Usability*. Master's Thesis. Freie Universität Berlin.
[WW01] Wang Y. ir Whitehead J. E. (2001). *International Accessibility of Open Source Software*. University of California, Santa Cruz.
[Wh56] Whorf B. L. (red.) (1956) *Language, Thought and Reality*. Selected Writings, MIT press, Cambridge, Mass.

# AUTORIAUS PUBLIKACIJŲ SĄRAŠAS

**Publikacijos, išspausdintos leidiniuose, įtrauktuose į pagrindinę Mokslinės informacijos instituto (ISI) duomenų bazę:**

1. **R. Laucius.** Issues of Selecting a Programming Environment for a Programming Curriculum in General Education. In: R. T. Mittermeir (Ed.), *Lect. Notes in Computer Science*, vol. 4226. Informatics Education – The Bridge between Using and Understanding Computers, 2006, p. 169–178.

**Publikacijos, išspausdintos tarptautinių konferencijų leidiniuose, įtrauktuose į Mokslinės informacijos instituto (ISI) duomenų bazę:**

2. **V. Dagienė, R. Laucius**. Internationalization of open source software: framework and some issues. In: T. Boyle, P. Oriogun, A. Pakstas (Eds.), *2nd International Conference Information Technology: Research and Education*, London: London Metropolitan University, 2004, p. 204–207.

**Publikacijos, išspausdintos periodiniuose ir tęstiniuose mokslo leidiniuose, registruotuose kitose tarptautinėse mokslinės informacijos duomenų bazėse**

3. **V. Dagienė, R. Laucius**. „*Free Pascal*“ panaudojimas informatikos kursui. *Lietuvos matematikos rinkinys*, 2001, 41 t., spec. nr., p. 267–271.
4. **R. Laucius, V. Dagienė**. Raštinės programinės įrangos „OpenOffice.org“ adaptavimas lokalės normoms. *Informacijos mokslai*, 26, 2003, p. 240–245.
5. **R. Laucius**. Hiperteksto rašyklių palyginimas lokalizavimo galimybių požiūriu. *Lietuvos matem. rink.*, 43 (spec. nr.), 2003, p. 254–25.
6. **R. Laucius**. Programinė įrangos vertimo specifika ir dalinis automatizavimas. *Lietuvos matem. rink.*, 44 (spec. nr.), 2004, p. 319–326.
7. **R. Laucius**. *Free Pascal* kompiliatoriaus internacionalizavimas. *Informacijos mokslai*, 34, 2005, p. 302–306.
8. **R. Laucius, V. Dagienė**. Lokalizavimo kurso projektavimas. Lietuvos matem. rinkinys, 45, spec. nr., 2005, p. 213–218.

**Publikacijos, išspausdintos kituose mokslo leidiniuose**

9. **R. Laucius.** Lokalės, jų sandara ir ypatumai. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, Kaunas: Technologija, 2003, p. I-(1–7).

# PRIEDAI

## 1 priedas. Darbe naudotų standartų santrumpų pilni pavadinimai

1. IEEE 1003 „Portable Operating System Interface“ (taip pat registruotas kaip ISO/IEC 9945)
2. ISO 15924 „Codes for the representation of names of scripts“
3. ISO 3166 „Codes for the representation of names of countries and their subdivisions“
4. ISO 4217 „Codes for the representation of currencies and funds“
5. ISO 639 „Codes for the representation of names of languages“
6. ISO 8601 „Data elements and interchange formats – Information interchange – Representation of dates and times“
7. ISO/IEC 10206:1991, ANSI/IEEE770X3.160-1989 „Programming Language Extended Pascal“
8. ISO/IEC 14652 „Specification method for Cultural Conventions“
9. ISO/IEC 14882 „Programming Languages — C++“
10. ISO/IEC 15897 „Procedures for registration for cultural conventions“
11. ISO/IEC 7185:1990 „Programming Language Pascal“
12. ISO/IEC 9899 „Programming language – C“
13. ISO/IEC 9945 „Information technology Portable Operating System Interface“
14. LST ISO/IEC 15897 „Informacijos technologija. Kultūros elementų registravimo procedūros“ (tapatus ISO/IEC 15897 „Information technology – Procedures for registration of cultural elements“)
15. RFC 3066 „Tags for the Identification of Languages“
16. RFC 3339 „Date and Time on the Internet: Timestamps“
17. UAX #11 „East Asian Width“
18. UAX #15 „Unicode Normalization“
19. UAX #21 „Case Mappings“
20. UAX #24 „Script Names“
21. UAX #29 „Text Boundaries“
22. UAX #31 „Identifier and Pattern Syntax“
23. UAX #34 „Unicode Named Character Sequences“
24. UAX #9 „The Bidirectional Algorithm“
25. UAX#14 „Line Breaking Properties“
26. UTR #16 „UTF-EBCDIC“
27. UTR #23 „Character Property Model“
28. UTR #25 „Unicode and Mathematics“
29. UTR #26 „Compatibility Encoding Scheme for UTF-16 8-Bit (CESU-8) “
30. UTR #30 „Character Foldings“
31. UTR #33 „Conformance Model“
32. UTR# 36 „Unicode Security Considerations“
33. UTR#17 „Character Encoding Model“
34. UTS #10 „Unicode Collation Algorithm“
35. UTS #18 „Unicode Regular Expressions“
36. UTS #22 „CharMapML“
37. UTS #35 „Locale Data Markup Language“
38. UTS #37 „Ideographic Variation Database“
39. UTS #6 „Compression Scheme for Unicode“

40. XPG4 „X/Open Portability Guide Issue 4“, taip pat žinomas kaip „Common Applications Environment Specification Issue 4“

2 priedas. Tyrimo skirto nustatyti populiariausius kompiliatorius rezultatai.

| **Programos pavadinimas** | | **Data** | **Kompiliatorius** |
|---|---|---|---|
| Nemokamos programos | | | |
| 1Fh Binary/Hex Editor 1.08 | | 2006 | Nepavyko nustatyti |
| Autorun Inf Editor 1.0 | | 2003 | MS Visual C++ |
| Brush Strokes Image Editor 1 | | 2004 | Borland C++ |
| CoffeeCup Free HTML Editor 8 | | 1996-2005 | Borland Delphi |
| Crimson Editor 3.7 | | 1999-2004 | MS Visual C++ |
| Davor's PHP Editor 1.0.3.3 | | 2003 | Borland Delphi |
| Dynamic HTML Editor v.1.9.6 | | 2003-2006 | Borland Delphi |
| EXPStudio Audio Editor Free 3.98 | | 2006 | MS Visual Basic |
| Free Hex Editor 3.12 | | 2005 | MS Visual C++ |
| GedLink Editor 1.3 | | 2003 | MS Visual Basic |
| HC Image Editor 2 2 | | 2005 | MS Visual Basic |
| HTML Editor 1.6.0.61 | | 2001 | Borland Delphi |
| MDB Browser and Editor 2.3 | | 2001 | MS Visual Basic |
| Microsoft Agent Character Editor 2.0 | | 1998 | MS Visual C++ |
| MP3 Tag Editor 3 | | 2004 | Borland Delphi |
| PageBreeze Free HTML Editor 3e | | 1999-2005 | MS Visual Basic |
| PC Image Editor 2.1 | | 2003-2006 | Nepavyko nustatyti |
| Peter's XML Editor 2.0 | | 2000-2003 | Borland Delphi |
| Power Tab Editor 1.7 build 80 | | 1996-2000 | MS Visual C++ |
| RealWorld Cursor Editor 2006.1 | | 2006 | Nepavyko nustatyti |
| RealWorld Cursor Editor 2006.1 | | 2003-2005 | Borland Delphi |
| VCW VicMan's Photo Editor 7.82 | | 1998-2004 | Borland Delphi |
| Wolf Web Editor Pro 3.0.3 | | 2003 | Borland Delphi |
| XRay XML Editor 0.9 build 11 | | 2001 | MS Visual Basic |
| ZS4 Video Editor 0.95 | | 2005 | MS Visual C++ |
| Laikinai nemokamos programos | | | |
| 3D Editor 2s | | 2005 | MS Visual Basic |
| All Editor 2.4.3 | | 1998-2003 | MS Visual C++ |
| Amazing Photo Editor 5.8 | | 2003 | Borland Delphi |
| Antechinus JavaScript Editor Standard 6.0 | | 2000-2005 | MS Visual C++ |
| Arabic Editor 3.5 | | 1999-2004 | Nepavyko nustatyti |
| Audio Editor Pro 2.21 | | 2000-2006 | MS Visual C++ |
| Audio Mp3 Editor 2.20 | | 2006 | MS Visual Basic |
| Boxer Text Editor 11.0.1 | | 2005 | Borland Delphi |
| Colorful Movie Editor 4.0 | | 2003 | Borland Delphi |

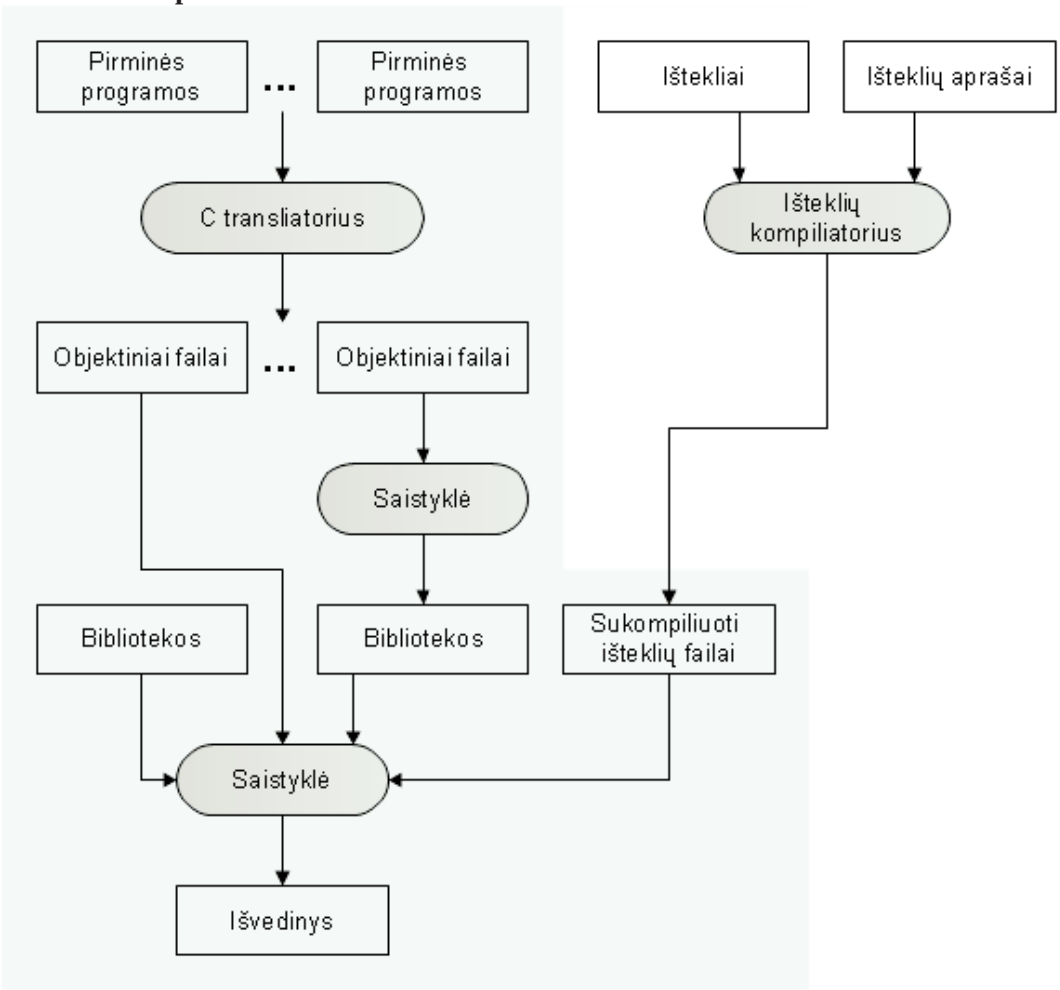
| | | | |
|---|---|---|---|
| Digital Audio Editor 7.1 | | 2000-2006 | Borland Delphi |
| DzSoft PHP Editor 4.0.0.6 | | 1999-2006 | Borland Delphi |
| Easy Photo Editor 1.9 | | 2004-2006 | MS Visual C++ |
| EasyHex Hex Editor 1.13 | | 2003 | MS Visual C++ |
| Fancy Movies Editor Pro 4.0 | | 2003 | Borland Delphi |
| FlexiMusic Wave Editor Nov2005 | | 1999-2005 | MS Visual Basic |
| Game Editor 1.3.5 | | 2006 | MS Visual C++ |
| Honestech Easy Video Editor 2.0 | | 2005 | MS Visual C++ |
| IconCool Icon Editor 5.26 build 60728 | | 1995-2006 | MS Visual Basic |
| Magic Audio Editor Pro 10.2.4 | | 2000-2006 | Borland Delphi |
| PDF Editor 2.4 | | 2002-2005 | Borland Delphi |
| PDFill PDF Editor 4.1 | | 2002-2006 | MS Visual C++ |
| Personal AVI Editor 1.57 | | 2001 | Nepavyko nustatyti |
| PHP Expert Editor 3.2.1 | | 2000-2004 | Borland Delphi |
| Sothink HTML Editor 2.5 | | 2002 | MS Visual C++ |
| WinHex Hex Editor 12.2 | | 2005 | Borland Delphi |

**Iš viso:**

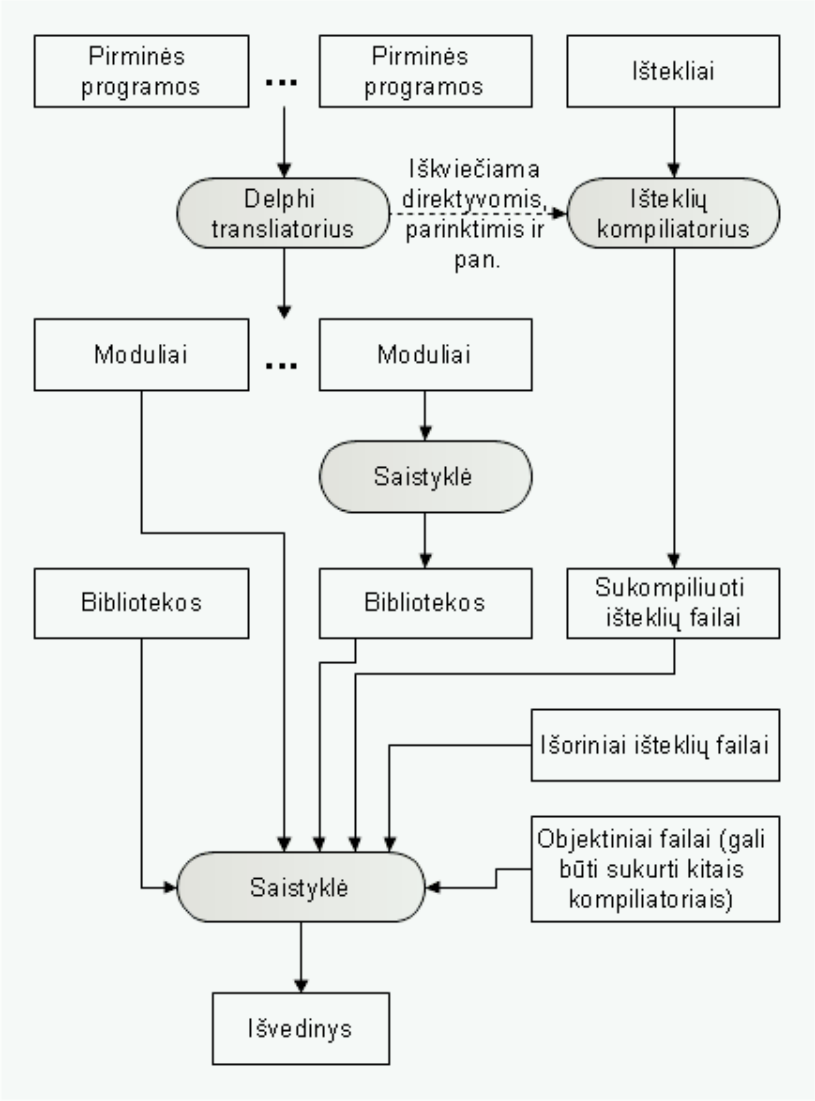
| **Kompiliatorius a** | **Kiekis** |
|---|---|
| Borland Delphi | 19 |
| MS Visual C++ | 15 |
| MS Visual Basic | 10 |
| Kiti | 6 |

# 3 priedas. Kompiliatorių veikimo schemos.

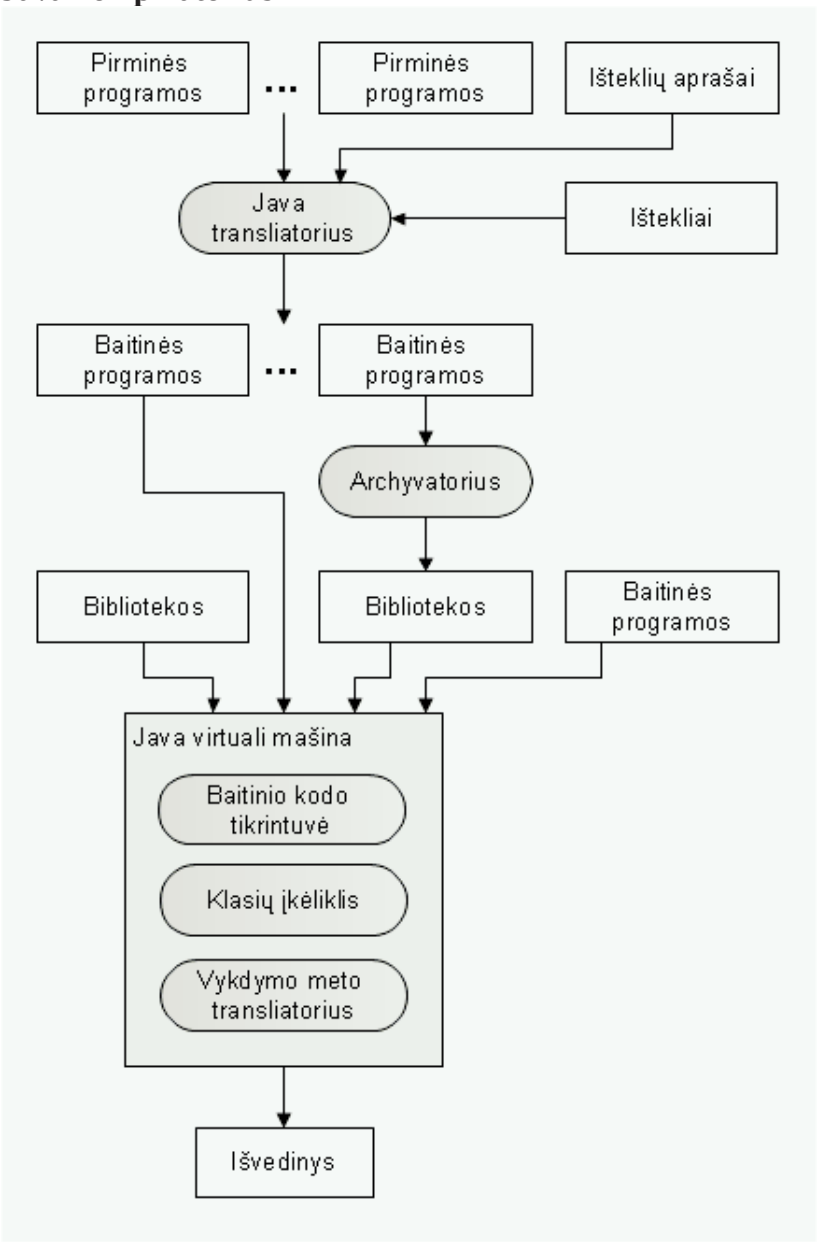
**C ir C++ kompiliatorius**



**Delphi kompiliatorius**



**Java kompiliatorius**



# 4 priedas. Kompiliatorių internacionalizacijos lygio tyrimo testų pavyzdžiai

1. Duomenų kodavimas
   - Bandoma sukompiliuoti programą, kompiliatoriaus parinktis nurodant Unikodu koduotame faile. Pavyzdžiui, jame gali būti panaudoti Unikodu koduoti failų vardai. Tikrinamas ar kompiliatorius teisingai supranta parinkčių failą.
   - Bandoma sukompiliuoti programą, kurioje panaudoti Unikodo ženklų turintys komentarai ir kiti elementai. Tikrinama ar kompiliatorius paprasčiausiai susidoroja su Unikodu koduotu pirminiu tekstu.
   - Ar išvedami duomenys koduojami Unikodu nustatoma pseudo-lokalizavus kompiliatorių. Kompiliatoriaus vykdymo metu įsitikinama ar jo išvedami duomenys koduojami Unikodu.
2. Išteklių atskyrimas
   2.1. Tekstinių išteklių atskyrimo realizacija
      - Ar tekstiniai ištekliai atskirti nuo veiksmų dalies nustatoma pasitelkiant lokalizavimui skirtas priemones pseudo-lokalizavimo būdu.
   2.2. Tekstinių išteklių atskyrimo realizacija
      - Sukompiliuojama programa, kurioje panaudotos išteklių eilutės (turinčios skaitinių parametrų, valdymo ženklų, komentarus, turinčios Unikodo ženklų). Tikrinama ar kompiliatorius atitinka reikalavimui. Tam gali būti naudojamos lokalizavimui skirtos priemonės.
   2.3. Kitų išteklių atskyrimas:
      - Ar tekstiniai ištekliai atskirti nuo veiksmų dalies nustatoma pasitelkiant lokalizavimui skirtas priemones pseudo-lokalizavimo būdu.Kitų išteklių atskyrimo realizacija:
      - Sukompiliuojama programa, kurioje panaudoti netekstiniai ištekliai. Tikrinama ar kompiliatorius atitinka pateiktus reikalavimus. Tam gali būti naudojamos lokalizavimui skirtos priemonės.
3. Įvedimas
   3.1. Įvedamų duomenų kodavimas
      - Programoje panaudota įvedimo paslauga. Duomenų įvedimui panaudoti eilutės ir simbolinis duomenų tipai (nes VMB jų įvedimas gali būti realizuojamas skirtingai). Sukompiliavus ir įvykdžius programą įvedant duomenis tiek iš ekrano, tiek iš failo tikrinama ar jie įvedimo metu koduojami teisingai ir nėra sugadinami perkoduojant.
   3.2. Įvedimo paslaugos realizacija
      - Programoje panaudota įvedimo paslauga. Sukompiliavus ir įvykdžius programą tikrinama ar duomenis galima įvesti naudojant įvairius įvedimo metodus ir jie įvedami teisingai.
4. Išvedimas
   4.1. Išvedamų duomenų kodavimas
      - Programoje panaudota teksto išvedimo paslauga. Teksto išvedimui panaudoti eilutės ir simbolinis duomenų tipai (nes VMB jų išvedimas gali būti realizuojamas skirtingai). Sukompiliavus ir įvykdžius programą tikrinama ar išvedimo paslaugos realizacija tenkina reikalavimus. Tikrinamas išvedimas ne tik į ekraną bet ir į failus.
   4.2. Teksto vaizdavimo paslauga
      - Programoje panaudota teksto vaizdavimo paslauga išvedant įvairių raštų tekstus, bei tekstus turinčius ženklų sudarytų iš kombinacinių sekų. Sukompiliavus ir įvykdžius programą tikrinama ar tekstas vaizduojamas teisingai.

- Pabandoma išplėsti jau sukompiliuotos programos teksto vaizdavimo paslaugą. Tikrinamas ar išplėtimas nėra labai sudėtingas ir nekelia problemų.

5. Teksto dorojimas
   5.1. Kodavimas
   - Programoje panaudota teksto perkodavimo paslauga atliekant perkodavimą iš Unikodo į keletą kitų koduočių ir atvirkščiai. Sukompiliavus ir įvykdžius programą tikrinama ar perkodavimas atliekamas teisingai.

   5.2. Ženklų savybės
   - Programoje panaudota ženklų savybių nustatymo paslauga atliekant įvairių raštų ženklų savybių nustatymą. Sukompiliavus ir įvykdžius programą tikrinama ar savybės nustatomos teisingai.

   5.3. Raidžių lygio keitimas
   - Programoje panaudota raidžių lygio keitimo paslauga atliekant įvairių raštų raidžių lygio keitimą įvairiose lokalėse nustatymą. Sukompiliavus ir įvykdžius programą tikrinama ar raidžių lygio keitimas atliekamas teisingai.

   5.4. Teksto skaidymas
   - Programoje panaudota teksto analizės paslauga atliekant įvairių raštų teksto skaidymą į grafemas, žodžius ir sakinius įvairiose lokalėse. Sukompiliavus ir įvykdžius programą tikrinama ar skaidymas atliekamas teisingai.

   5.5. Teksto normalizavimas
   - Programoje panaudota teksto normalizavimo paslauga atliekant teksto normalizavimą pagal visas 4 formas. Sukompiliavus ir įvykdžius programą tikrinama ar normalizavimas atliekamas teisingai.

   5.6. Dvikrypčio teksto transformavimas
   - Programoje panaudota dvikrypčio teksto transformavimo paslauga atliekant teksto transformavimą iš loginės tvarkos į vaizdavimo tvarką ir atvirkščiai. Sukompiliavus ir įvykdžius programą tikrinama ar transformavimas atliekamas teisingai.

6. Kultūriniai elementai
   6.2. Datos ir laikas
   - Programoje panaudota datų ir laiko paslauga. Išbandomas datų pervedimas įvairių kalendorių atžvilgiu, bei laiko pervedimas įvairių laiko juostų atžvilgiu. Tikrinama ar datos ir laikas keičiami teisingai.

   6.3. Formatavimas ir analizė
   - Programoje panaudota formatavimo ir analizės paslauga, formatuojant ir analizuojant datas ir laiką, skaičius, pinigines sumas pasirenkant įvairias lokales. Tikrinama ar formatavimas ir analizė atliekami teisingai.

   6.4. Pranešimų formatavimas
   - Programoje panaudota pranešimų formatavimo paslauga. Formatuojant pranešimus eilutės jungiamos tarpusavyje, į pranešimus įterpiami datos ir laikas, skaičiai, piniginės sumos pasirenkant įvairias lokales. Tikrinama ar pranešimų formatavimas atliekami teisingai ir atitinka reikalavimus.

   6.5. Rikiavimas, palyginimas ir paieška
   - Programoje naudojama rikiavimo paslauga tekstinio sąrašo rikiavimui. Rikiavimas atliekamas pasirenkant įvairias lokales, atsižvelgiant ir neatsižvelgiant į raidžių lygį. Tikrinama ar rikiavimas atliekamas teisingai.
   - Programoje naudojama palyginimo paslauga teksto eilučių palyginimui. Palyginimas atliekamas pasirenkant įvairias lokales, atsižvelgiant ir neatsižvelgiant į raidžių lygį. Tikrinama ar palyginimas atliekamas teisingai.

- Programoje naudojama paieškos paslauga ieškant teksto fragmento teksto masyve. Paieška atliekama pasirenkant įvairias lokales, atsižvelgiant ir neatsižvelgiant į raidžių lygį. Tikrinama ar paieška atliekamas teisingai.
- Pabandoma išplėsti jau sukompiliuotos programos sąrašų rikiavimo paslaugą. Tikrinamas ar išplėtimas nėra labai sudėtingas ir nekelia problemų.

6.6. Kompiliatoriaus realizacija

- Programa sudaryta taip, kad išprovokuotų reikalavimuose numatytus kompiliatoriaus veiksmus. Pavyzdžiui, laikas įterpiamas išvedant informaciją apie programos kompiliavimo trukmę. Rikiavimas atliekamas kompiliuojant programą turinčią eksportuojamų funkcijų. Ir t.t.

7. Leksikos elementai

7.1. Vardai

- Programoje naudojami vardai sudaryti taip, kad išprovokuotų galimas kompiliatoriaus klaidas, susijusias su vardų sintaksės neatitikimu Unikodo standartui.
- Programoje panaudoti Unikodo ženklų turintys vardai, dalis jų eksportuojami. Sukompiliavus programą tikrinama ar objektas teisingai apdoroja Unikodu koduotus vardus, įterpia derinimui skirtą informaciją. Pavyzdžiui, tuo galima įsitikinti naudojant programų derinimui skirtas priemones. Taip pat tikrinama ar teisingai koduojami eksportuojamų funkcijų vardai. Pavyzdžiui, tuo galima įsitikinti atvėrus sukompiliuotą programą įprasta teksto žiūrykle.

7.2. Eilutės

- Programoje panaudotos Unikodo ženklų turinčios eilutės bei simboliai. Atliekamos standartinės eilučių operacijos (konstantų priskyrimas, kintamųjų priskyrimas, priskyrimas tarp skirtingo tipo eilučių, eilučių jungimas, skirtingų tipų eilučių jungimas, simbolio ir eilutės jungimas ir t.t.). Tikrinama ar objektas teisingai supranta ir apdoroja Unikodu koduotas eilutes bei simbolius.

7.3. Baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai.

- Ar šiuos elementus galima lokalizuoti įsitikinama pseudo-lokalizavimo būdu.

7.4. Skaičiai

- Programoje panaudoti skaičiai sudaryti iš kituose raštuose naudojamų dešimtainių skaitmenų. Tikrinama ar kompiliatorius juos teisingai supranta ir apdoroja.

8. Failų sistemos paslaugos

- Bandoma sukompiliuoti programų, kurią sudarančių failų varduose panaudoti Unikodo ženklai.
- Bandoma sukompiliuoti programą, jos kompiliavimui naudojamas parinktis nurodant parinkčių faile, kurio varde panaudoti Unikodo ženklai.
- Bandoma sukompiliuoti programą, kurioje direktyvų pagalbą įterpiami arba nurodomi failai, kurių varduose panaudoti Unikodo ženklai.
- Programoje panaudojamos VMB funkcijos skirtos darbui su failų sistema. Sukompiliavus ir įvykdžius programą tikrinama ar ji teisingai supranta Unikodo ženklų turinčius failų vardus.

# 5 priedas. Kompiliatorių internacionalizacijos lygio tyrimo rezultatai

| | **Delphi (.NET)** | **Delphi (Win32)** | **Visual C++** | **Java** |
|---|---|---|---|---|
| Duomenų kodavimas | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Išteklių atskyrimas | | | | |
| Kompiliatoriaus išteklių atskyrimas | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Tekstinių išteklių atskyrimo realizacija | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Kitų išteklių atskyrimo realizacija | Taip | Taip | Nepakankamai | Nepakankamai |
| Įvedimas | | | | |
| Įvedamų duomenų kodavimas | Nepakankamai | Ne | Nepakankamai | Nepakankamai |
| Įvedimo paslaugos realizacija | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Išvedimas | | | | |
| Išvedamų duomenų kodavimas | Nepakankamai | Ne | Nepakankamai | Nepakankamai |
| Išvedimo paslaugos realizacija | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Teksto dorojimas | | | | |
| Kodavimas | Nepakankamai | Nepakankamai | Nepakankamai | Taip |
| Ženklų savybės | Ne | Ne | Nepakankamai | Taip |
| Raidžių lygio keitimas | Nepakankamai | Nepakankamai | Nepakankamai | Taip |
| Teksto skaidymas | Ne | Ne | Nepakankamai | Nepakankamai |
| Teksto normalizavimas | Ne | Ne | Ne | Ne |
| Dvikrypčio teksto transformavimas | Ne | Ne | Ne | Taip |
| Kompiliatoriaus realizacija | Nepakankamai | Nepakankamai | Taip | Nepakankamai |
| Kultūriniai elementai | | | | |
| Kultūrinių elementų įkėliklis | Nepakankamai | Ne | Nepakankamai | Taip |
| Datos ir laikas | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Formatavimas ir analizė | Nepakankamai | Ne | Nepakankamai | Taip |
| Pranešimų formatavimas | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Rikiavimas, palyginimas ir paieška | Nepakankamai | Ne | Nepakankamai | Taip |
| Kompiliatoriaus realizacija | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |
| Leksikos elementai | | | | |
| Vardai | Nepakankamai | Nepakankamai | Ne | Taip |

| | | | | |
|---|---|---|---|---|
| Eilutės | Taip | Nepakankamai | Nepakankamai | Taip |
| | **Delphi (.NET)** | **Delphi (Win32)** | **Visual C++** | **Java** |
| Baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai. | Ne | Ne | Ne | Ne |
| Skaičiai | Ne | Ne | Ne | Ne |
| Skyryba | Ne | Ne | Ne | Ne |
| Failų sistema | Nepakankamai | Nepakankamai | Nepakankamai | Nepakankamai |