PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Valentina Dagienė
Gintautas Grigas
Tatjana Jevsikova

Matematikos ir informatikos institutas

PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Valentina Dagienė
Gintautas Grigas
Tatjana Jevsikova

Matematikos ir
informatikos
institutas

UDK 004.41
Da77

Monografiją spaudai rekomendavo
Matematikos ir informatikos instituto taryba, 2010 m. gegužės 26 d. Posėdžio protokolas Nr. 4

Recenzentai:
prof. habil. dr. *Genadijus Kulvietis*, Vilniaus Gedimino technikos universitetas
doc. dr. *Vladas Tumasonis*, Vilniaus universitetas

ISBN 978-9986-680-47-5

Turinys

Įvadas

Kompiuteriai dabar naudojami kasdien, paplitę visame pasaulyje ir padeda atlikti pačius įvairiausius darbus. Kompiuterių veiksmingumas labai priklauso nuo programinės įrangos. Viena iš svarbių kultūriniu ir ekonominiu požiūriais programinės įrangos savybių yra jos sąsajos su naudotoju pateikimas naudotojo kalba: nepriklausomai nuo to, kur programa sukurta, ji turi būti patogi naudotojui, tarsi būtų sukurta jo kultūrinėje terpėje.

Programinės įrangos pritaikymas (arba adaptavimas) tam tikrai kalbos ir kultūros aplinkai vadinamas lokalizacija, o pats procesas – lokalizavimu. Lokalizuota programinė įranga turi taisyklingai rengti ir apdoroti ta kalba parašytus dokumentus, vartoti tai kalbai tinkamą koduotę, toje kalboje ir valstybėje priimtus matavimo vienetus ir jų žymenis, datos, laiko ir kitus formatus, į tą kalbą išverstus naudotojo matomus tekstus (meniu, dialogo langų užrašus, kompiuterio pranešimus), dokumentaciją (elektroninius ir popierinius žinynus, vadovėlius). Taip pat turi būti naudojama tai kultūrinei terpei būdinga simbolika, spalvos ir kiti elementai. Visa, kas čia išvardyta, paprastai apibrėžiama lokalėje.

Poreikis lokalizuoti programinę įrangą atsirado tada, kai pradėta masiškai eksportuoti programas į įvairias valstybes. Šiandien, augant kompiuterių ir interneto naudotojų skaičiui, šis poreikis vis didėja. Todėl lokalizavimo tyrinėjimai ir jo kokybės gerinimo būdų paieška – aktuali problema.

Programinės įrangos lokalizavimo pradžioje (XX a. 9-ame dešimtmetyje) pagrindinis dėmesys buvo skiriamas galimybei apdoroti lokalės tekstus. Tik vėliau buvo imta nagrinėti programinės įrangos pritaikymą įvairioms kultūrinėms ir kalbinėms normoms, galiojančioms konkrečioje kalboje ar teritorijoje, dar vėliau – visapusiškam programinės įrangos adaptavimui lokalei, tarsi programa būtų specialiai suprojektuota konkrečios vietovės bendruomenei. Dar ir dabar tenka pastebėti, kad

kartais programinės įrangos lokalizavimas kartais suprantamas kaip vertimas. Tačiau net ir vien programinės įrangos tekstų pateikimas kita kalba labiau panašus ne į įprastą, bet į gerokai adaptuotą vertimą.

Daugelis visuotinai naudojamų programų dar nėra pakankamai lokalizuotos, pasitaiko įvairių klaidų ir neatitikimų kalbos normoms.

Programinės įrangos lietuvinimas – tai atskiras programų lokalizavimo atvejis, kai programa modifikuojama taip, kad tiktų darbui lietuviškoje kalbinėje ir kultūrinėje terpėje. Lokalizacija gali būti įvairaus lygio: pradedant programos pritaikymu kurios nors kalbos rašmenims apdoroti ir baigiant visišku programos parengimu kalbinei ir kultūrinei terpei, kad su ja dirbantysis jaustųsi taip, lyg programa būtų sukurta toje terpėje. Taigi lietuvinimą derėtų suprasti kiek bendresne prasme.

Šioje knygoje išsamiai aptarsime programinės įrangos lokalizavimo eigą, pagrindinius komponentus, įvardysime problemas, pateiksime siūlymų, aptarsime kultūrines dimensijas. Nors programinės įrangos sąvoka skiriasi nuo kompiuterių programų sąvokos, knygoje šiuos terminus vartosime sinonimiškai, nes lokalizavimas susijęs su bet kuria kompiuterio programa: ar tai būtų visa sistema, įvardijama programine įranga, ar pavienė programa, kompiuterio naudotojo atžvilgiu visos programos turi būti lokalizuotos.

Knygą sudaro devyni skyriai. Pirmajame skyriuje išsamiai aptariami programinės įrangos lokalizavimo ir internacionalizavimo problematika, terminija, supažindinama su lokalizacijos laipsniais.

Antrajame skyriuje supažindinama su kalbomis ir jų rašto ženklais, ženklų rikiavimu, identifikavimu.

Trečiasis skyrius skirtas ženklų kodavimui. Trumpai apžvelgiama kodavimo istorija pradedant ikikompiuterinių laikų kodavimu (Morzės abėcėle), aptariamos šešių ir septynių bitų koduotės. Daugiausiai dėmesio skiriama dabar naudojamoms aštuonių bitų koduotėms ir unikodui.

Ketvirtajame skyriuje gilinamasi į lokalę, aptariami jos pagrindiniai komponentai – kultūros elementai, pristatomi įvairūs lokalės modeliai.

Penktajame skyriuje nagrinėjami lokalizuojamieji ištekliai, išsamiai pristatomi formaliųjų ir atributinių gramatikų taikymo metodai.

Šeštajame skyriuje aptariamas lokalizavimo procesas: nuo parengiamųjų darbų, programų adaptavimo, dialogų tekstų vertimo, visų tekstų redagavimo iki kompleksinio lokalizuotų programų testavimo ir jų darbų tolesnio tęstinumo.

Septintajame skyriuje trumpai pristatomos lokalizavimo priemonės: žodynai, vertimo atmintys, lokalizavimo programos.

Aštuntajame skyriuje aptariami specifiniai lokalizavimo atvejai. Tai programų kalbų ir jų kompiliatorių lokalizavimo ir internacionalizavimo klausimai. Taip pat trumpai apžvelgiamas interneto svetainių lokalizavimas.

Devintajame skyriuje aptariamos kultūrinės dimensijos.

Tikimės, kad monografijoje pateikta medžiaga bus naudinga lokalizavimo problemas tyrinėjantiems mokslininkams. Knyga turėtų dominti ir lokalizuotojus praktikus – pagelbėtų geriau suvokti lokalizavimo procesus, našiau ir sistemingiau atlikti lokalizavimo darbus.

Monografijos autoriai dėkoja visiems, kurie talkino suteikdami informaciją, konsultuodami, diskutuodami terminų klausimais. Džiaugiamės kolegų Anitos Juškevičienės, Viktoro Dagio, Agnės Strelkauskytės ir Aido Žandario atidžiu monografijos nagrinėjimu ir naudingomis pastabomis, taip pat Rimanto Žakausko parengtomis kodų lentelėmis ir kitais paveikslais.

Autoriai nuoširdžiai dėkoja Matematikos ir informatikos instututo direktoriui prof. Gintautui Dzemydai už patarimus, recenzentams Vilniaus Gedimino technikos universiteto prof. Genadijui Kulviečiui ir Vilniaus universiteto doc. Vladui Tumasoniui – už kruopščią rankraščio peržiūrą.

1. KOMPIUTERIŲ PROGRAMŲ LOKALIZAVIMAS IR INTERNACIONALIZAVIMAS

Programinės įrangos lokalizavimo ir internacionalizavimo tyrimai prasidėjo XX amžiaus devintajame dešimtmetyje. Nuo to laiko paskelbta nemažai mokslinių ir metodinių publikacijų iš programinės įrangos lokalizavimo ir internacionalizavimo sričių.

Pradinius lokalizavimo tyrimus vykdė stambiosios programinės įrangos projektavimo kompanijos: „Sun Microsystems“, „Microsoft“ ir „Apple Computers“ (Nielsen, 1990; Microsoft Corporation, 1990; Apple Computer, 1992). Vėliau ši sritis tapo konkurencine, todėl informacija apie lokalizaciją paprastai buvo prieinama tik tokių kompanijų viduje ir neskelbiama viešai. Lygiagrečiai su pramonine veikla vyko akademiniai tyrimai, skirti programinei įrangai projektuoti taip, kad ji tiktų įvairioms kalboms ir kultūroms (pvz., Russo, Boor, 1993; Teasley ir kt., 1994; Evers, Day, 1997).

Programinės įrangos gamintojų publikuojama medžiaga dažniausiai demonstravo konkretaus produkto projektavimo ir lokalizavimo gerąją patirtį. Vėliau, paplitus įvairiems mobiliesiems įrenginiams ir internetui, domėjimasis lokalizavimo ir internacionalizavimo darbais padidėjo. Nemažai dėmesio skiriama atvirajai internetinei programinei įrangai lokalizuoti (Dagienė, Grigas, 2006; Dagienė, Laucius, 2004; Nichols ir kt., 2005; Lanier, 2005; Laucius, Dagienė, 2003; Frimannsson, Hogan, 2005 ir kt.). Iš programinės įrangos lokalizavimo ir internacionalizavimo sričių parašyta keletas daktaro disertacijų (Atkin, 2001; O'Sullivan, 2001; Evers, 2001a; Laucius, 2007; Gaspari, 2007; Jevsikova, 2009), nuo 1995 metų Airijos Limeriko universiteto Lokalizavimo tyrimų centras kasmet rengia tarptautines lokalizavimo konferencijas[1], su lokalizavimu ir internacionalizavimu susiję tyrimai pristatomi žmogaus ir kom-

[1] Annual Localisation Conference, http://www.localisation.ie

piuterio sąveikos konferencijų darbuose (pvz., Reinecke, Bernstein, 2007; Schadewitz, Jachta, 2007; Wan Mohd Isa, Md Noor, 2007; Ford, Gelderblom, 2003).

Programinės įrangos lokalizavimą dauguma autorių apibrėžia kaip „programinės įrangos pritaikymą tam tikrai kalbinei ir kultūrinei terpei“ (Uren ir kt., 1993; Dagienė ir kt., 2008; Esselink, 2000 ir kt.). Tačiau, kaip rodo konkrečių lokalizuotų programų analizė, ši sąvoka suprantama nevienareikšmiškai. Įvairūs autoriai suteikia skirtingas prasmes žodžiui „pritaikymas“ ir frazei „kalbinė ir kultūrinė terpė“. Vieni (autoriai, lokalizuotojai ar programuotojai) tai supranta tik kaip programos tekstų ir žinynų vertimą, kiti – tik kaip vertimą ir adaptavimą kai kurioms kultūrinėms normoms (pvz., datos ir laiko formatai, dešimtainės trupmenos ir tūkstančių skirtukai, koduotės ir pan.).

Todėl reikėtų įvesti bent dvi sąvokas: dalinė programos lokalizacija ir visiška programos lokalizacija. Visiška programos lokalizacija laikysime programos formos ir turinio pritaikymą tarsi ji būtų specialiai sukurta tam tikrai kalbinei ir kultūrinei terpei, kuriai lokalizuojama (Schäler, 2002). Tačiau dažniau būna atliekama tik dalinė lokalizacija.

Įvairių autorių pateikiami programinės įrangos internacionalizacijos apibrėžimai šiek tiek skiriasi. Publikacijose dažnai remiamasi Lokalizacijos industrijos standartų asociacijos LISA[1] pateiktu apibrėžimu: internacionalizacija yra produkto apibendrinimo procesas tam, kad jis galėtų apdoroti (palaikyti) skirtingas kalbas ir kultūrines nuostatas jo neperprojektavus. Apibendrinę galėtume sakyti, kad programinės įrangos internacionalizacija – tai programinės įrangos ir jos duomenų struktūrų projektavimas taip, kad visa tai būtų galima lengvai adaptuoti įvairioms kalboms ir kultūroms.

Šioje knygoje internacionalizacijos sąvoką vartosime dvejopai: 1) programinės įrangos savybių, darančių ją lengvai adaptuojamą įvairioms kalboms ir kultūroms, visumai nusakyti; 2) dalykui, nagrinėjančiam metodus, priemones, technologijas, naudojamas sudarant programinės įrangos adaptavimo įvairioms kalboms ir kultūroms sąlygas, įvardinti.

Internacionalizacijos ir lokalizacijos raida vyko beveik kartu. Ji siejasi su programinės įrangos gamintojų noru gauti daugiau pelno išplečiant savo produktų rinkas į kitas šalis. Pradžioje kompiuterių programomis daugiausia naudojosi tik kompiuterių specialistai, tad programinės įrangos rinka buvo nedidelė. Siekiant įtikti kitų šalių

[1] The Localisation Industry Standards Association, http://www.lisa.org/

pirkėjams užtekdavo išversti programų dokumentaciją ir tai buvo laikoma jos lokalizavimu. Visa programinė įranga buvo verčiama tik pavieniais atvejais.

Paplitus asmeniniams kompiuteriams, programinės įrangos lokalizavimo poreikis išaugo. Naudotojai ėmė teikti pirmenybę lokalizuotai programinei įrangai, nes ji buvo praktiškesnė, darbas su ja efektyvesnis. Lokalizavimas buvo pradėtas nuo pačių būtiniausių dalykų – programinės įrangos pritaikymo apdoroti duomenis, atitinkančius nacionalinius standartus, kultūrines nuostatas, raštą. Dialogo kalba į kitas kalbas buvo ne visada verčiama (Esselink, 2003).

Iš pradžių lokalizavimas buvo atliekamas pirminio programos teksto lygmeniu. Tačiau netrukus toks metodas pasirodė esąs labai neefektyvus ir nepatikimas. Reikalingas išversti teksto eilutes, išbarstytas po visą programų tekstą, būdavo sunku surasti. Taisyti, kompiliuoti ir derinti tekdavo kelis skirtingų lokalių programinės įrangos 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 – atsirado internacionalizavimo sąvoka.

Internacionalizacijos atsiradimo pradžioje pagrindiniai uždaviniai buvo: sudaryti galimybę programinės įrangos 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. Plėtojantis programinei įrangai, internacionalizacijos samprata kito ir jai keliamų uždavinių daugėjo.

Vienas iš pirmųjų dažnai cituojamų šaltinių, turėjusių didelę įtaką programinės įrangos internacionalizacijos sampratos raidai buvo Russo ir Boor straipsnis (Russo, Boor, 1993). Jame teigiama, kad siekiant programinės įrangos paklausos tarptautinėje rinkoje, jos gamintojai turi susipažinti su tarpkultūriniais skirtumais ir į juos atsižvelgti, tuo pačiu pakeisdami iki tol gyvavusį programinės įrangos gamybos procesą – į jos gamybą įtraukdami internacionalizavimą. Pažymima, jog visiškai internacionalizuotos programinės įrangos gamybos procesas neturi apsiriboti vien teksto vertimo galimybių sudarymu, skirtingais datų, laiko, skaičių formatais ir panašiais elementais. Pateikiamas nuo kultūros priklausomų programinės įrangos elementų (susijusių su tekstų vaizdavimu, formatais, paveikslais, spalvomis, programos funkcionalumu, ir pan.) sąrašas su komentarais. Taip pat svarbu, kad programinės įrangos gamybos ciklas papildomas internacionalizacijos testavimu prieš išleidžiant pradinę jos versiją. Anksčiau to nebuvo daroma.

Russo ir Boor buvo vieni pirmųjų, kurie atkreipė dėmesį į tai, kad programinės įrangos praktiškumui įtakos turi daugybė nuo kultūros priklausomų programinės

įrangos veiksnių. Vėliau Yeo (Yeo, 1996) šiuos kultūrinius veiksnius suskirstė į dvi grupes: akivaizdžius ir neakivaizdžius. Akivaizdūs kultūriniai veiksniai (pvz., matavimo vienetai, kalendoriai) pakankamai gerai apibrėžti ir lengvai suprantami. Tuo tarpu neakivaizdūs veiksniai (pvz., grafinė, akustinė ir vizualinė informacija, spalvos, funkcionalumas, metaforos) iki šių dienų lieka nepakankami apibrėžti arba iš viso neapibrėžti, nors yra svarbūs kuriant internacionalizuotą programinę įrangą.

Kiti šaltiniai (Gribbons, 1997) pateikia kiek kitokią programinės įrangos internacionalizacijos klasifikaciją: skirstoma į du lygmenis – paviršinį ir kultūrinį. Paviršinis lygmuo susijęs su akivaizdžiais kultūriniais veiksniais, yra pakankamai aiškus ir santykinai nesunkiai realizuojamas, giluminis lygmuo, susijęs su spalvų, paveikslų ir kitų mažiau formalių elementų pritaikymu, kelia internacionalizavimo ir lokalizavimo problemų, nes tie elementai nėra pakankamai gerai ištirti ir apibrėžti. Shen šiuos lygmenis vaizduoja kaip ledkalnį, kurio paviršinis lygis yra matomas ir pakankamai aiškus, tačiau pagrindinę ir kur kas mažiau išnagrinėtą internacionalizacijos dalį sudaro kultūrinis lygis (Shen, 2000).

Tiriant kultūros įtaką programinės įrangos praktiškumui buvo nustatyta ir daugiau internacionalizacijai svarbių kultūrinių veiksnių. Evers disertacijoje (Evers, 2001a) pateiktas tyrimas, kurio tikslas įvertinti naudotojo kultūrinės patirties įtaką programinės įrangos sąsajai suvokti. Nustatyta, kad metaforiškai išreikštus programinės įrangos sąsajos elementus skirtingų kultūrų respondentai gali asocijuoti su savo aplinkos objektais, kurie neatitinka gamintojo užkoduotų objektų ir dėl to gali nukentėti programų praktiškumas. Tyrimas taip pat parodė, kad tam pačiam tikslui pasiekti skirtingose kultūrose gali būti priimta naudoti skirtingus veiksmus. Todėl programų internacionalizavimas neturi apsiriboti vien tik galimybių programinės įrangos sąsajai lokalizuoti sudarymu, bet gali apimti ir programų atliekamus veiksmus (apie tai pamiršta daugelis autorių, rašančių apie internacionalizaciją). Kersten su bendraautoriais (Kersten ir kt., 2000) įvardija keletą programinės įrangos tipų, kuriems tai gali būt ypač svarbu, bei pateikia idėjų, kaip tokiais atvejais gali būti realizuojama programų internacionalizacija.

Choong ir Salvendy (Choong, Salvendy, 1998) tyrimas pademonstravo, jog internacionalizuojant programas tikslinga atsižvelgti į skirtingose kultūrose įprastus suvokimo ir mąstymo stilius. Šio tyrimo metu nustatyta, kad Kinijos kompiuterių naudotojų (jiems labiau būdingi konkretaus suvokimo ir tematinio mąstymo stiliai) ir JAV naudotojų (būdingi abstraktaus suvokimo ir funkcinio mąstymo stiliai) programinės įrangos naudojimo efektyvumas skiriasi, kai informacija pateikiama abs-

trakčia arba konkrečia išraiška, o sąsaja realizuojama funkciniu arba tematiniu būdu. Rezultatai parodė, kad naudotojai efektyviau naudojasi ta programine įranga, kuri atitinka jų suvokimo ir mąstymo stilius.

Net keletas tyrimų (Ford, Gelderblom, 2003; Marcus, Gould, 2000) visiškai arba iš dalies patvirtina Hofstede (Hofstede, 1991) apibrėžtų kultūrinių dimensijų (galios santykio, individualizmo prieš kolektyvizmą, vyriškumo prieš moteriškumą, neapibrėžtumų vengimo, ilgalaikės perspektyvos prieš trumpalaikę) reikšmingą įtaką programinės įrangos praktiškumui. Pavyzdžiui, Marcus ir Gould (Marcus, Gould, 2000) analizuoja skirtingoms kultūroms atstovaujančių šalių tinklalapius ir pateikia tai patvirtinančių pavyzdžių.

Keičiantis supratimui apie kultūros įtaką programinės įrangos praktiškumui, kito ir internacionalizacijos samprata. Iš pradžių buvo laikoma, kad programų internacionalizuotumas reiškia jos neutralumą kultūrinių nuostatų atžvilgiu, t. y. lokalizuojant tokią programinę įrangą turi pakakti tik jos vertimo. Tačiau, kaip pažymima Kersten tyrime (Kersten ir kt., 2000), nėra universalios kultūros arba jos pagrindų, todėl negalima sukurti programinės įrangos, kuri savaime būtų pritaikyta visų kultūrų atstovams. Anksčiau minėti tyrimai patvirtino, kad visiškas programų adaptavimas konkrečiai kultūrai turi reikšmingą teigiamą įtaką jos praktiškumui. Tuo tarpu šis internacionalizavimo būdas praktiškumą aukoja vardan mažesnių investicijų, todėl jis laikytinas ydingu (Vatrapu, 2002).

Viena iš internacionalizacijos sampratų remiasi idėja, kad internacionalizuota programinė įranga turi pati automatiškai prisitaikyti prie konkrečios kultūrinės terpės nuostatų. Tačiau sukurti programinę įrangą, kuri visiškai prisitaikytų, sunku, nes sunku formaliai apibrėžti visų nuo kultūros priklausomų programinės įrangos elementų. Pavyzdžiui, sunku formaliai apibrėžti paveikslus 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 nėra duomenų bazių, kuriose būtų sukaupti išsamūs įvairių kultūrų nuostatų aprašai bei programų internacionalizavimo metodų, kurie leistų pasiekti aukštą tokios internacionalizacijos lygį.

Lokalizacijos kokybei pastaruoju metu keliamus reikalavimus neformaliai galima apibūdinti posakiu: „programinė įranga turi būti adaptuota tam tikrai kalbinei ir kultūrinei terpei taip, lyg ji būtų sukurta toje terpėje“. Tai pasiekti galima tik adaptavus programinę įrangą atsižvelgiant į visas konkrečioje terpėje galiojančias kalbines ir kultūrines nuostatas, todėl programų internacionalizavimo uždavinys – sudaryti visas galimybes, kad toks adaptavimas būtų įmanomas.

Literatūroje lietuvių kalba galima pastebėti vartojamus terminus lokalizavimas ir lokalizacija. Šiame darbe naudosime terminą *lokalizavimas*, kai turimas omenyje programinės įrangos adaptavimo kultūrinei terpei procesas, o terminą *lokalizacija* – kai kalbama apie adaptavimo kultūrinei terpei proceso rezultatą arba kai įvardijama programinės įrangos lokalizuota atmaina.

1. Daugelis autorių (O'Sullivan, 2001; Esselink, 2000; Schäler, 2002; Yang, 2007 ir kt.) nurodo, kad programinės įrangos lokalizavimo darbus galima suskirstyti į dvi stambias dalis: Programos adaptavimas, kad ji teisingai veiktų konkrečioje kalbinėje ir kultūrinėje terpėje (teisingų koduočių, skaičių ir pinigų sumų formatų, datų ir laiko formatų, pirmosios savaitės dienos ir kt. vartojimas).
2. Dialogo tekstų (meniu, dialogo langų užrašų, įvairių pranešimų, žinyno ir naudotojo vadovo ir kitų, susijusių dokumentų) vertimas ir adaptavimas.

Lokalizavimo įgyvendinimo kokybę įtakoja tai, kiek programinė įranga yra pritaikyta lokalizavimui. Internacionalizavimas daugelio autorių apibrėžiamas kaip programinės įrangos projektavimo dalis, pritaikant programinę įrangą tarptautinei rinkai (Dagienė ir kt., 2004, Hogan ir kt., 2004, ir kt.). Kuo geriau programinė įranga yra internacionalizuota, tuo paprasčiau ją lokalizuoti. Šios dvi sąvokos yra glaudžiai susijusios ir dažnai mokslinėje literatūroje vartojamos greta. Internacionalizavimas yra laikomas programinės įrangos produkto elementų atskyrimu į kultūriniu požiūriu neutralią ir nuo kultūros priklausomas dalis (Carey, 1998). Todėl programinės įrangos analizė lokalizavimo požiūriu yra kartu ir jos internacionalizacijos tyrimas, kai lokalė nefiksuojama, arba dalinis tyrimas, kai fiksuojama viena ar daugiau lokalių.

Nuo programų internacionalizavimo lygio priklauso jų lokalizavimo sąnaudos: kuo daugiau programa internacionalizuota, tuo lengviau ją lokalizuoti. Senesnių programų internacionalizavimo lygis dažniausiai būna žemesnis (anksčiau dar nebuvo suvokta internacionalizavimo reikšmė).

Kai kurie autoriai, pavyzdžiui, Taylor (Taylor, 1992), laiko, kad lokalizavimas reikalauja gerokai daugiau pastangų, negu internacionalizavimas, kadangi lokalizuotojams tenka spręsti ir tas problemas, kurios nebuvo išspręstos internacionalizavimo metu.

Lietuviškai vartojami du terminai: internacionalizavimas ir internacionalizacija. Šiame darbe naudosime terminą *internacionalizavimas*, kai turimas omenyje programinės įrangos projektavimo procesas, o terminą *internacionalizacija* – kai 1) kalbama apie programinės įrangos projektavimo proceso rezultatą arba 2) programinės įrangos savybių, darančių ją lengvai adaptuojama įvairioms kalboms ir kultūroms, visumą.

1.1. Programų lietuvinimo pradžia

Lokalizuoti programinę įrangą Lietuvoje buvo pradėta vėliau negu daugelyje kitų Europos šalių.

Pirmoji operacinė sistema buvo sulietuvinta 1996 metais – tuomet IBM bendrovė, vykdydama Lietuvos mokyklų kompiuterizavimo projektą, į lietuvių kalbą išvertė operacinę sistemą OS/2.

Bendrovė „Microsoft“ lietuvinti savo programas pradėjo 2001 metais. Pirmiausia iš dalies buvo sulietuvinta tekstų rengyklė „Word XP“ (2001 m.), vėliau – kitos raštinės paketo „Office 2003“ programos (2004 m.). 2002 metais iš dalies (pritaikyta darbui daugiakalbėje terpėje) sulietuvinta operacinė sistema „Windows XP Professional“, o 2007 metais – „Windows Vista“. 2009 m. spalio 22 d. išleista „Windows 7“ operacinė sistema buvo visiškai sulietuvinta, ji pirmą sykį išleista vienu metu su originalia angliška versija.

Operacinę sistemą „Linux“ ir jos terpėje veikiančias programas pradėjo lietuvinti atvirųjų programų judėjimo dalyviai. Buvo lietuvinama „Gnome“, KDE, „Debian“, „Mandrake“ (vėliau – „Mandriva“) sąsajos. Šie darbai tęsiami iki šiol – lietuvinamos nuolat atnaujinamos versijos. Daugiausia sulietuvinta „Gnome“ sąsaja.

Nuo 1995 metų programų lietuvinimo darbai vykdomi Matematikos ir informatikos instituto Programavimo metodologijos skyriuje (dabar – Informatikos metodologijos skyrius). Buvo orientuojamasi į mokyklose naudojamų programų lietuvinimą, nagrinėjami tokios programinės įrangos poreikiai ir taikymas mokymui (Dagienė, 1998; Dagienė, 2000).

Visų lietuvinimo darbų tikslas – aprūpinti galutinį kompiuterio naudotoją lietuviškomis programomis. Todėl Matematikos ir informatikos instituto mokslininkai stengėsi lietuvinti labiausiai reikalingas programas, rūpinosi jų kokybe, priežiūra, naujų versijų lietuvinimu. Lokalizacijos veikla buvo suprantama plačiąja prasme – ne vien kaip lokalizuotos programos gamyba, bet ir prieš ją einantys parengiamieji darbai ir paskesnė lokalizuotos programos priežiūra, įskaitant ir naujų originalios programos versijų lokalizavimą (lokalizacijų atnaujinimą).

Parengiamieji programų lietuvinimo darbai – parinkti labiausiai reikalingas ir pačias geriausias programas, parengti tokį lietuvinimo planą, kad sulietuvintos programos kokybė būtų kuo aukštesnė.

Pirmoji Matematikos ir informatikos institute sulietuvinta programa – elektroninio pašto programa „Demos Mail“, veikianti operacinėje sistemoje DOS. Kito pasirinkimo

nebuvo, kadangi tuo metu dominavo Atviros Lietuvos fondo remiama elektroninio pašto sistema, veikianti UUCP protokolu, kuriai buvo pritaikyta tik ši programa.

Pradėjus plisti „Windows“ sistemai reikėjo kitos pašto programos. Galimi kandidatai buvo „Eudora“, „Pegasus Mail“ ir naršyklėje „Netscape Navigator“ (vėliau pavadintoje „Netscape Communicator“) esanti elektroninio pašto programa.

„Eudora“ buvo iš karto atmesta dėl to, kad ji į išsiunčiamo laiško antraštę įrašydavo koduotę ISO-8859-1, nors laiškas būdavo pateikiamas operacinėje sistemoje esančia koduote (mūsų atveju – „Windows-1257“). Tais laikais nedaug pašto programų mokėjo šia informacija pasinaudoti. Ja nesinaudojo ir pati „Eudora“. Dėl šios priežasties „Eudora“ buvo pripažinta netinkama naudoti Lietuvoje.

Buvo ilgokai svyruojama tarp dviejų programų: „Pegasus Mail“ ir iš „Netscape Navigator“ išsirutuliojusios programos „Mozilla“. Apie 1998 metus „Pegasus Mail“ buvo viena geriausių elektroninio pašto programų, o „Mozillos“ pirmtakas „Netscape Communicator“ buvo gera interneto naršyklė su gana vidutine tinklalapių rašykle ir prasta elektroninio pašto dalimi. Todėl iš pradžių buvo planuota lokalizuoti abi programas, kad būtų padengtos dvi reikalingiausios darbo internete sritys – saityno naršymas ir paštas.

Pirmenybė buvo teikiama gerai pašto programai „Pegasus Mail“, be to, tuo metu „Mozilla“ praktiškai dar neveikė – buvo tik viltis, jog kada nors ji atsigaus. Tačiau su „Pegasus Mail“ autoriumi nesisekė bendradarbiauti. Dėl neaiškių priežasčių jis atidėliojo sutarties pasirašymą, o kai ji buvo pasirašyta, teko ilgai laukti lokalizuojamųjų išteklių pirminių tekstų. Autoriui sunkiai sekėsi taisyti nurodytas originalios programos klaidas, susijusias su tinkamų šriftų bei koduočių parinkimu, – ką nors pataisius atsirasdavo naujų klaidų. Nesisekė ir su pasiūlymais tobulinti programą: koduotę pasirinkti iš sąrašo (o ne rinkti jos pavadinimą), įtraukti UTF-8 koduotę. Laikui bėgant aiškėjo, kad „Pegasus Mail“atsilikimas nuo kitų pašto programų didėja, programa sensta. Todėl „Pegasus Mail“ pašto programa taip ir liko nesulietuvinta.

Tuo metu „Mozillos“ būklė buvo daug prastesnė. Tačiau ši programa sparčiai tobulėjo. Optimistiškai nuteikė ir jos pirmtako „Netscape Communicator“ autorių operatyvi reakcija dėl koduotės ISO-8859-13 įtraukimo į programą (tos pačios problemos sprendimas bendrovėje „Microsoft“ užtruko dvejus metus). Turint tai omeny ir nemažai rizikuojant buvo pasirinkta „Mozilla“. Dabar galima tvirtinti, kad pasirinkimas buvo teisingas. Per keletą metų, „Mozillos“ pašto programa „Thunderbird“ pralenkė ne tik tuo metu populiarią „Pegasus Mail“, bet ir kitas programas, o naršyklė pasivijo „Internet Explorer“. Sulietuvinta programa įgijo ir naujų savybių – lie-

tuvišką adresyną, numatytąsias paieškos sistemas, visas lietuviškiems rašto ženklams vartojamas koduotes, jau nekalbant apie lietuvišką sąsają.

Panašiai, tik trumpesniu keliu buvo pasirenkamos ir kitos lokalizuojamos programos: „Comenius Logo“ konkuravo su „Microworlds“, „Geometer‘s Sketchpad“ – su „Cabri“, „Total Commander“ – su FAR, išteklių tvarkytuvė „LeechGet“ – su keturiomis tokios pat paskirties programomis.

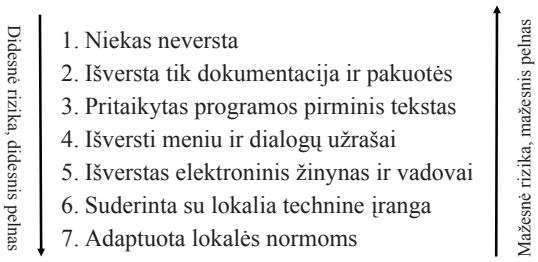
1.2. Programinės įrangos lokalizavimo laipsniai

Poreikis lokalizuoti programinę įrangą padidėjo, kai pradėta masiškai eksportuoti programas į įvairias valstybes. Augant kompiuterių ir interneto naudotojų skaičiui, šis poreikis vis didėja.

Lokalizuoti programas pradėta palaipsniui. Lokalizavimo tematikos straipsnių autoriai išskiria kelis lokalizavimo laipsnius. Kai kurie autoriai (pavyzdžiui, Kano, 1995; Carey, 1998) pagal atliktus lokalizavimo darbus skiria septynis lokalizavimo laipsnius ir sieja juos su rizikos lygiu ir pelnu (1 pav.).

Kuo daugiau išvardytų darbų atlikta, tuo geresnis lokalizavimas. Autoriai pažymi, kad tam tikrais atvejais gali būti naudinga lokalizuoti tik iš dalies (pvz., jei programa yra skirta inžinieriams, programuotojams, serverių administratoriams, tai nebūtina lokalizuoti visą programos dokumentaciją).

Kiti autoriai skiria lokalizavimo laipsnius pagal jų istorinį įgyvendinimą. Pavyzdžiui, lokalizavimo plitimo pradžioje buvo apsiribojama galimybe apdoroti lokalės tekstus, vėliau pradėta pritaikyti programas prie įvairių kultūrinių ir kalbinių normų, galiojančių konkrečioje kalboje ar teritorijoje, dar vėliau – visapusiškai adaptuoti



1 pav. Lokalizavimo laipsniai

programinę įrangą lokalei, tarsi programa būtų specialiai suprojektuota tai lokalei. Šioje monografijoje išskirsime tris pagrindinius lokalizavimo laipsnius:

1. Lokalės tekstų apdorojimas.
2. Adaptavimas lokalės normoms.
3. Visapusiškas lokalizavimas.

Pirmasis laipsnis – galimybė apdoroti lokalės tekstus. Vakarų Europoje tai buvo atliekama iš karto, kai tik kompiuterius pradėta naudoti tekstinei informacijai apdoroti, nes programa, negalinti apdoroti tos kalbos rašto ženklų, yra bevertė. Tuo tarpu Vidurio Europos ir Rytų Europos valstybėse dėl žemo jų ekonominio išsivystymo lygio ir nelegalios programinės įrangos paplitimo šis procesas vėlavo. Užuot bent iš dalies lokalizavus įrangą, ilgą laiką buvo taikstomasi su svetimų rašmenų naudojimu savos kalbos tekstams užrašyti. Pavyzdžiui, rusai rašė lotynų abėcėlės raidėmis, lietuviai raides su diakritiniais ženklais keitė kitomis raidėmis. Dabar šios problemos techniškai jau išspręstos. Tačiau sprendimų vėlavimas pagimdė socialines ir kultūrines problemas. Daugelis kompiuterių naudotojų prisitaikė prie rašto ženklų ignoravimo ir anksčiau egzistavusio ribojimo vartoti visus kalbos ženklus kompiuteryje ir nebesinaudoja elementariomis šiuolaikinių kompiuterių ir programinės įrangos galimybėmis. Todėl reikalingas visuomenės švietimas.

Antrasis laipsnis – programinės įrangos pritaikymas prie įvairių kultūrinių ir kalbinių normų, galiojančių konkrečioje kalboje ar teritorijoje (valstybėje, įvairiakalbės valstybės autonominėje dalyje ir pan.): skaičių ir valiutos užrašymo formatai, laiko ir datos formatai, religinių ar kitokių simbolių vartojimo reglamentavimas, specifinių simbolių ir spalvų simbolinės reikšmės, tipinės mandagumo frazės. Kai kurių kultūrinių reikalavimų nepaisymas gali būti rimtų klaidų priežastimi (pvz., neteisingi dešimtainės trupmenos ir tūkstančių skirtukai) arba programą daryti svetimą susiklosčiusioms kultūrinėms tradicijoms. Tokių reikalavimų gana daug. Svarbiausi jų pateikiami kalbų lokalėse (žr. 4 skyr.). Reikia pripažinti, kad dar nemažai programų gamintojų savo gaminiuose nenumato tokių reikalavimų ir jų reguliavimo arba lokalizavimo galimybių. Todėl programų lokalizuotojams tikslinga veikti dviem kryptimis: rasti būdus, kaip lokalizuoti šiuos programų elementus, o programinės įrangos gamintojus informuoti apie tai, kad jie naujas programų laidas geriau pritaikytų lokalizavimui.

Trečiasis laipsnis – visapusiškas programinės įrangos adaptavimas. Šiuolaikinį kompiuterį galima laikyti intelektualiu prietaisu, atliekančiu aptarnavimo funkcijas. Pavyzdžiui, tekstų rengyklė pakeičia ikikompiuterinių laikų mašininkę, elektroninio

pašto programa – laiškanešį ir t. t. Akivaizdu, kad aptarnaujantis personalas privalo mokėti kliento kalbą. Kai kompiuteris pakeičia žmogų, jis privalo „mokėti“ jo aptarnaujamo kliento kalbą – priimti kliento kalba pateikiamas komandas ir klientui teikti informaciją kliento kalba, t. y. visi tekstai, matomi kompiuterio ekrane, turi būti rodomi kliento kalba. Tokių tekstų yra gana daug. Todėl visų jų vertimas ir adaptavimas reikalauja nemažai sąnaudų ir istoriškai buvo pradėtas vėliau. Dabartiniu metu tai užima didžiausią lokalizavimo darbų dalį ir visame pasaulyje vyksta labai intensyviai. Šį lokalizavimo laipsnį galima laikyti galutiniu lokalizavimo etapu, nes norint jį įgyvendinti reikalingi pirmieji du. Tai pats reikšmingiausias etapas. Todėl toliau kalbėdami apie lokalizavimą turėsime omeny trečiąjį laipsnį kaip galutinį ir problematiškiausią.

Internetinėje programinėje įrangoje, ypač veikiančioje serveryje, galima siekti trečiojo laipsnio (visapusiško) lokalizavimo. Yra įprasta, kad lokalizuojant interneto svetainę, lokalizuojamas tik tas turinys, kuris matomas svetainės lankytojui, pvz., tinklalapio vaizdas naršyklės lange, bet pamirštama apie svetainės tinklalapių pirminį tekstą, išteklių universaliuosius adresus, rodomus naršyklės adresų juostoje. Pavyzdžiui, Yijun Yu, Jianguo Lu ir kt. (2005) pateikia metodą, kaip lokalizuoti ne tik XML dokumentų turinį, bet ir gaires. Naudojant SGML standartą (ISO 8879), kuriame naudojamas turinio ir gairių kodavimas unikodu, kuriami gairių vertimų žodynai ir XSLT stiliai gairėms konvertuoti, „XPath“ reiškiniai žodyno užklausoms formuoti. Visa tai leidžia sukurti atitinkamus priedus internetinei programinei įrangai, pavyzdžiui, naršyklei, tinklalapių rengyklei, kurie leistų matyti tinklalapio pirminį tekstą lokalės kalba.

Skirtingi lokalizavimo laipsniai gali būti nevienareikšmiško lokalizavimo termino supratimo priežastimi, pavyzdžiui, žemas įvairių programų lokalizacijų lygis paskatino kai kuriuos autorius netgi taikyti alternatyvųjį metodą, analogišką kino filmų subtitrams (Lepouras, Weir, 2003). Programos kiekvienos komandos, su kuria susiduria naudotojas, vertimas pateikiamas dinamiškai (etiketėje). Tai gali tikti daugiakalbėje kultūrinėje aplinkoje, tačiau neišsprendžia problemų, su kuriomis susiduria lokalizuotojai, o tik atitolina nuo jų sprendimo.

Pirmajam, antrajam ir iš dalies trečiajam lokalizavimo laipsniams įgyvendinti reikalinga informacija – programinėje įrangoje naudojami elementai, kurie yra skirtingi įvairioms kalboms ir kultūroms. Ši informacija yra struktūrizuojama, formalizuojama ir kaupiama naudojantis lokalių modeliais (jie nagrinėjami 4-ame skyriuje).

Trečiajam lokalizavimo laipsniui realizuoti reikalinga informacija, kuri priklauso nuo gilesnių kultūrinių tradicijų, mąstymo ir suvokimo ypatumų, pateikiama remiantis suformuluotomis kultūrinėmis dimensijomis.

Trečiojo lokalizavimo laipsnio negalima pasiekti, jeigu programos ištekliai, kurie nėra neutralūs kalbiniu ir kultūriniu požiūriu (pvz., visi dialogų tekstai, grafika, garsai), nebus atskirti nuo programos pirminio teksto. Todėl 5 skyriuje analizuojami egzistuojantys tokių išteklių atskyrimo metodai ir lokalizuojamųjų išteklių pateikimo formatai.

2. KALBOS IR RAŠTO ŽENKLAI

Vienas didžiausių žmonijos išradimų, skirtų bendravimui – abėcėlė. Ją sudaro įvairūs ženklai, vartojami pranešimams išreikšti. Ženklas – kokio nors rinkinio elementas, turintis sutartinę reikšmę ir skirtas bendravimui, atliekantis komunikacinę funkciją. Bendru atveju ženklu galime laikyti ir sutartinį paveikslėlį – piktogramą, hieroglifą, bet kurios abėcėlės raidę, skaitmenį.

Kol žmonija sukūrė raštą, turėjo praeiti daug laiko. Maždaug prieš 5–6 tūkstančius metų, tose vietose, kur žmonijos civilizacija buvo pasiekusi aukštesnį negu kitur lygį, prie Nilo, Tigro ir Eufrato upių, ėmė formuotis ideografiniai (lot. „ideo“ – mintis, gr. „grapho“ – rašau) rašto tipai. Kalba buvo grindžiama morfemomis – nekaitomomis šaknimis. Ideografinis raštas tapo raidžių, rašto ženklų formavimosi pradžia[1]. Viena seniausių pasaulio abėcėlių – finikiečių (mus pasiekęs dokumentas, rašytas finikiečių raidėmis, datuojamas X a. pr. Kr.). Semitai ir aramėjai jį patobulino – atsirado arabų raštas.

Raidės iš Azijos į Europą per Viduržemio jūrą persikėlė finikiečių laivais. Jos ne iš karto pritapo, mat, finikiečiai, beje, kaip ir egiptiečiai, naudojo tik priebalses, o balses praleisdavo. Europiečiams reikėjo ir balsių.

Manoma, kad iš finikiečių pirmieji raides perėmė graikai, jie net išsaugojo raidžių pavadinimus. Pirmasis graikų raidynas turėjo 24 ženklus: 17 buvo skirta priebalsėms ir 7 balsėms žymėti. Oficialų statusą šis raidynas įgijo 403 m. pr. Kr., kai jį pripažino Atėnai. Iš Graikijos raidės pradėjo sparčiai plisti po visą Europos žemyną.

Lotyniškoji abėcėlė atsirado etruskams perėmus ir pritaikius graikiškąją – ir dar buvo tobulinama keletą šimtmečių. Lotyniškoji abėcėlė paplito kartu su krikščionybe, ne tik Europoje, bet ir visame pasaulyje. Dabar ja naudojasi daugiau kaip trečdalis

[1] Apie senuosius raštus galima paskaityti R. Neimanto knygelėje „Raštas: nuo piktogramos iki raidės“, Kaunas: „Orientas“, 1998, o apie pasaulio kalbas – V. Mažiulio knygoje „Pasaulio tautų kalbos“, Vilnius, „Gimtasis žodis“, 2001.

planetos gyventojų, pritaikę savo reikmėms – vieni sumažinę, kiti padidinę ženklų skaičių. Iš įdomesnių pastebėjimų: havajiečiams užtenka 16 raidžių, latviams reikia net 33. Lietuviai nedaug atsilieka – mūsų abėcėlę sudaro 32 raidės.

Šitiek trumpai apie rašto istoriją – mums rūpi šiandien vartojamos abėcėlės, jų ypatumai. Lokalizuojant programas pirmiausia reikia gerai įsigilinti į abiejų kalbų – iš kurios ir į kurią verčiama programa – ypatumus, susipažinti su problemomis, apžvelgti esamas rekomendacijas.

2.1. Raštai

Raštas – tai rašmenų rinkinys kartu su rašybos taisyklėmis, reikalingas tam tikros kalbos ar kalbų grupės tekstams užrašyti. Skirtingose kultūrose kalbai užrašyti vartojami įvairūs ženklai bei taisyklės, todėl egzistuoja didelė raštų įvairovė.

Pagal tai, kaip rašto (rašytinės kalbos) elementai atitinka šnekos (garsinės kalbos) elementus, raštus galima suskirstyti į tris stambias grupes: raidinius, skiemeninius ir ideografinius.

Raidiniuose raštuose vieną garsą atitinka viena arba kelios raidės. Jeigu vieną garsą atitinka viena raidė, tai tokia kalba vadinama fonetine. Fonetinių kalbų pavyzdžiai – čekų, sanskritas. Lietuvių, latvių, rusų, ukrainiečių kalbos taip pat laikomos fonetinėmis, nors jose yra tarimo išimčių. Nefonetinės kalbos pavyzdys – anglų kalba.

Skiemeninėse kalbose vieną skiemenį žymi vienas rašto ženklas. Būdingas pavyzdys – japonų katakanos ir hiraganos rašmenys.

Ideografinėse kalbose vieną sąvoką (žodį) žymi vienas ženklas (ideograma). Pavyzdžiai – kinų, japonų, korėjiečių kalbos.

Kompiuteryje naudojamus rašmenis apibrėžia standartas ISO 15924 (šiuo metu jame yra 115 rašmenų kodai). Aptarsime pagrindinius raštų skirtumus, turinčius įtakos jų realizacijai kompiuteryje.

Unikodo standarte pateikiama raštų klasifikacija, sudaryta iš penkių grupių (The Unicode Standard, 2006):

1. Abėcėlės.
2. Abdžadai[1].

[1] Abdžado ir abugido pavadinimai sukurti jungiant tiems raštams tipiškų abėcėlių pirmąsias raides.

3. Abugidai.
4. Skiemeniniai.
5. Logo-skiemeniniai.

Šią klasifikaciją pasiūlė raštų sistemų ekspertai P. T. Daniels ir W. Brigzt (1996). Ji pagrįsta raštų sandaros savybėmis, t. y. kaip kalba išreiškiama raidėmis. Skirtinguose raštuose raidės gali reikšti priebalsius, balsius, skiemenis, žodžius arba jų dalis ir sąvokas. Suprantama, kad nuo to priklauso ir raštuose naudojamų ženklų skaičius (žymėti priebalsiams jų reikia sąlyginai nedaug, tuo tarpu žymėti žodžiams jų reikia daugybės). Trumpai apžvelgsime šias raštų grupes ir kitus programinės įrangos lokalizavimui ir internacionalizavimui svarbius raštų skirtumus.

Abėcėlės. Raidės skirstomos į balses ir priebalses pagal tai, kokie garsai jomis užrašomi: balsiai ar priebalsiai. Balsės ir priebalsės turi vienodą vertę. Abėcėlinių raštų grupei priskiriami lotynų, kirilicos, graikų, gruzinų, runų ir kiti raštai. Labiausiai paplitęs lotynų raštas. Jį vartoja šimtai kalbų, taip pat lietuvių.

Šios grupės raštų raidės (kaip abdžadų, abugidų ir skiemeninių raštų) daugiau ar mažiau atitinka kalbos garsus. Raštai, kuriuose raidės (arba ženklai) atitinka kalbos garsus vadinami fonetiniais. Tam, kad būtų galima tiksliai užrašyti įvairių kalbų garsus buvo sukurta tarptautinė fonetinė abėcėlė (IPA, 1999; Hansmeyer, 2002).

Daugelyje kalbų ryšys tarp raidžių ir garsų yra netiesioginis, jis apibrėžiamas formaliomis taisyklėmis, bet pastarosios gali turėti įvairių išimčių, kaip, pavyzdžiui, anglų kalboje (Roach, 2000). Kalbos garsams išreikšti priebalsėmis ir balsėmis pakanka nedaug raidžių, todėl dauguma kalbų 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. Šiai grupei taip pat priklauso hebrajų, sirų ir kiti raštai. Jie turi panašų raidžių skaičių, kaip ir abėcėlės.

Abugidai. Abugiduose raidės reiškia priebalsius ir balsius. Tačiau skirtingai nuo abėcėlių 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 šiuos prijungtus balsius linkę vadinti angliškų terminu „alphasyllabic“ pabrėždami būtent šį jų bruožą (Brigzt, 1996; Sprogt, 2003). Abugidus daugiausia naudoja Pietų Azijos šalys. Šiai grupei priklauso devanagari, bengalų, tamilų, tibetiečių ir daug kitų raštų. Iš jų labiausiai paplitęs devanagari raštas, naudojamas sanskrito, hindi ir daugelio kitų kalbų.

Vienas svarbiausių šių raštų bruožų yra tai, kad raidžių forma ir netgi jų išsidėstymo tvarka[1] 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 abugidams). Daugeliu atveju raidės gali neatpažįstamai pakeisti savo formą, lyginant su pirminių sujungtų raidžių forma. 2-ame paveiksle pateikiamas tokio raidžių junginio pavyzdys. Šių raštų realizacijos kompiuteryje dar vadinamos sudėtinėmis.

2 pav. Devanagari raidžių junginio pavyzdys

Kaip ir prieš tai minėtų grupių raštai, abugidai taip pat nepasižymi dideliu raidžių skaičiumi. 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 reiškia skiemenis. Galima išskirti du šios grupės raštų pogrupius: 1) paprastieji – kuriuose raidės vaizdavimas neturi ryšio su jos reiškiamo skiemens fonetine sandara (šiam pogrupiui priklauso hiragana, katakana, bopomofo, čerokių ir kt. raštai), 2) būdingieji – kuriuose minėtas ryšys egzistuoja (jam priklauso etiopų, hangulas ir kt. raštai).

Nors šie raštai priklauso tai pačiai grupei, tačiau raidžių skaičiumi jie gali labai skirtis. Pavyzdžiui, hiragana ir katakana turi apytikriai[2] po 100 raidžių, etiopų raštas – apie 300, hangulas – apie 11000 (šio rašto raidės sudarytos iš fonetinių ženklų, kurių yra 24, sekų).

Logo-skiemeniniai raštai. Šiuose raštuose raidės reiškia žodžius arba žodžių dalis. Kai kurie autoriai, rašydami apie šiuos raštus, vartoja terminus *ideografinis*, *logografinis*, *piktografinis*, labiau paryškindami atskiras šių raštų savybes, tuo tarpu terminas logo-skiemeninis apima visas šias savybes, todėl yra tikslesnis. Pagrindinis šios grupės raštas yra hanų, vartojamas kinų, japonų ir korėjiečių[3,4]. Jis

[1] Tai neturi nieko bendra su dvikrypčių raštų raidžių dėstymo tvarka.

[2] Raidžių skaičius rašomas apytikriai todėl, kad dažnai dalis raidžių nevartojamos (pasenusios), nors formaliai jos egzistuoja, todėl dažnai sunku pasakyti tikslų raidžių skaičių. Be to, šiuose pavyzdžiuose jis nėra aktualus.

[3] Šiai grupei žymėti anglų kalboje dažnai vartojama CJK santrumpa, reiškianti: China, Japan, Korea.

[4] Anksčiau šį raštą naudojo ir vietnamiečiai, tačiau dabar jie yra perėję prie paprastesnės, lotyniškos rašto sistemos.

turi gana daug raidžių, pavyzdžiui, kinų žodyne[1], išleistame 1716 metais, jų pateikta virš 47 tūkstančių.

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 jo skirtumų. Be to, moderninant raštą buvo sukurta nauja, supaprastinta jo forma, kuri ženklų vaizdavimo būdu iš esmės skiriasi nuo tradicinės. Dėl šių priežasčių labai padidėjo šiuose raštuose naudojamų raidžių skaičius.

Reikia paminėti dar keletą esminių raštų skirtumų, kurie turi didelę reikšmę programų lokalizavimui ir internacionalizavimui (Hall, Hudson, 1997):

1. Rašymo kryptis.
2. Priklausomybė nuo konteksto.
3. Skyryba.
4. Apipavidalinimas.
5. Paieška ir rikiavimas.
6. Eilučių laužymas.
7. Skaitmenys.

Trumpai juos apžvelgsime, išsamiau apie kelis kitus bus kalbama 2.3 skyriuje. Daugiau informacijos apie šiuos skirtumus galima rasti unikodo standarte, tačiau jame nesiekiama išsamiai aprašyti visų skirtumų, jie pateikiami daugiausia kaip pavyzdžiai, į kuriuos turi būti atsižvelgta gaminant su šiuo standartu suderinamą programinę įrangą.

Rašymo kryptis. Pagal tai raštai skirstomi į dvikrypčius ir vienkrypčius. Vienkrypčiuose raštuose dažniausiai rašoma iš kairės į dešinę (pavyzdžiui, lotynų), dvikrypčiuose – iš dešinės į kairę su intarpais, rašomais priešinga kryptimi. Pavyzdžiui, arabų rašte arabų raidės yra rašomos iš dešinės į kairę, o skaičiai – priešinga kryptimi, kitų raštų ženklais parašyti intarpai – iš kairės į dešinę. Prie vienkrypčių taip pat priskiriami raštai, kuriuose rašoma vertikaliai (pvz., tradicinis kinų raštas).

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 raštuose, pavyzdžiui, arabų (Hall, Hudson, 1997).

Skyryba. Skirtinguose raštuose gali būti vartojami skirtingi skyrybos ženklai. Pavyzdžiui, lotynų ir daugelyje kitų raštų žodžius priimta skirti tarpais, bet tailandiečiai

[1] Daugiau informacijos apie šį žodyną galima rasti: http://en.wikipedia.org/wiki/Kangxi_Dictionary.

žodžių skyrybos ženklų nevartoja. Taip pat reikia turėti omenyje, kad skirtingose kultūrose, net ir naudojančiose tą patį raštą, gali būti naudojami skirtingi skyrybos ženklai, pavyzdžiui, kabutės (žr. 2.3.1 skyr.).

Apipavidalinimas. Skirtingose kultūrose ir raštuose yra nusistovėjusios įvairios taisyklės, kaip turėtų būti apipavidalinamas tekstas. Pavyzdžiui, vienose kultūrose priimta svarbius užrašus išskirti kursyvu, kitose – didesniais tarpais tarp raidžių ar pusjuodžiu šriftu. Apipavidalinimui taip pat priklauso antraščių rašymas, sąrašų ir išnašų žymėjimas bei daugelis kitų nuostatų. Raštų savybės taip pat lemia dalį šių skirtumų. Tai, kas nesunkiai realizuojama viename rašte, gali būti sunkiai realizuojama ar net neįmanoma realizuoti kitame.

Paieška ir rikiavimas. Paieškos ir rikiavimo skirtumus lemia ne tik skirtingose kultūrose nusistovėjusios nuostatos, tačiau ir raštų skirtumai. Pavyzdžiui, abėcėlės 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: hiragana, katakana ir kandži, todėl japonų paieškos arba rikiavimo taisyklės yra sudėtingesnės negu daugelio kitų kalbų.

Eilučių laužymas. Taisyklės, susijusios su eilučių sandara, pavyzdžiui, žodžių skaidymas, kėlimas į kitą eilutę, skirtingose kultūrose skiriasi.

Skaitmenys. Mes vartojame arabiškus skaitmenis: 1, 2, 3, ... Tačiau arabai skaitmenis žymi kitokiais ženklais: ١, ٢, ٣, ... Kiti skaitmenų pavyzdžiai: devanagari: १, २, ३,..., tajų: ๑, ๒, ๓,.... Kinai turi du skaitmenų variantus, vieną įprastiems skaičiams, kitą – pinigų sumoms.

2.2. Europos kalbų grupės

Mus domina, kaip lokalizuoti programas į lietuvių kalbą, tačiau norėtųsi kiek plačiau apsidairyti ir suvokti, kaip mūsų raštas atrodo tarp kitų, kaip klasifikuojamos kalbos, kokie jų abėcėlių sudarymo principai ir pan. Pasiremsime Europos kalbų vadovu bibliotekininkams (Allen, 1995), t. y. pasinaudosime jo sistemine kalbų informacija.

Istorijos eigoje kalbų ir jomis šnekančių žmonių skaičius nuolatos kinta. Susidarius nepalankioms sąlygoms, vienos kalbos išnyksta, vietoj jų įsigali kitos, dažniausiai tos, kurioms palankios aplinkybės leidžia išplėsti vartojimo sritį. Kalbos skiriasi ne tik savo paplitimo laipsniu ir funkcine apimtimi, bet ir savo struktūra, raiškos priemonių sudėtimi bei pobūdžiu. Dažniausiai būna taip, kad vieną kurią nors kalbą vartoja vienos tautybės žmonės. Tačiau esama tokių atvejų, kai viena vartojama kelių tautybių ir net kelių valstybių. Pavyzdžiui, anglų kalba yra valstybinė ne tik Jungtinėje Karalystėje, bet ir Jungtinėse Amerikos Valstijose, Australijoje, Kanadoje (išskyrus Kvebeko sritį, kurioje kalbama prancūziškai), Naujojoje Zelandijoje, Indijoje ir kt.

Pagal genealoginę klasifikaciją, sudarytą XIX amžiuje lyginamosios istorinės kalbotyros atstovų, kalbos skirstomos į šeimas, šeimos į šakas, o šakos į grupes. Šaka – stambiausias šios klasifikacijos vienetas, jungiantis visas iš vienos prokalbės kilusias kalbas. Pavyzdžiui, išskiriamos indoeuropiečių, ugrų-finų, tiurkų-totorių ir kitos kalbų šeimos.

Šaka yra mažesnis genealoginės klasifikacijos vienetas, apimantis dar artimesnes kalbas giminaites negu šeima. Pavyzdžiui, iš indoeuropiečių šeimos kalbų išskiriamos baltų, slavų, germanų ir kitos šakos. Dar mažesnis vienetas – grupė. Tai pačiai grupei priklausančių kalbų artumo laipsnis paprastai būna gana didelis, nors ir nevienodas imant atskiras grupes. Pavyzdžiui, tarp rusų, baltarusių ir ukrainiečių kalbų, sudarančių rytų slavų grupę, yra daug bendrumų, tačiau tarp vokiečių ir anglų kalbų, įeinančių į vakarų germanų grupę, skirtumų yra žymiai daugiau.

Kalbos, kurių gramatinės reikšmės daugiau reiškiamos afiksais, vadinamos sintetinėmis (lietuvių, lenkų, rusų ir kt.), o reiškiamos tarnybiniais žodžiais – analitinėmis (anglų, prancūzų, bulgarų ir kt.). Tačiau kalbininkai pastebi, kad pasaulyje nėra nei absoliučiai grynų sintetinių, nei analitinių kalbų: sintetinėse kalbose greta afiksų gramatinės reikšmės neretai dar reiškiamos ir tarnybiniais žodžiais, o analitinėse – greta tarnybinių žodžių ir tam tikrais afiksais. Apie kalbas daugiau paskaityti galima knygose (Golovinas, 1982; Palionis, 1985; Zinkevičius, 1983).

Aptarsime tik dalį Europos kalbų, būtent, indoeuropiečių kalbų šeimos germanų, romanų, slavų ir baltų šakas ir keletą ugro-finų kalbų (1–5 lentelės). Visas kalbas nagrinėsime kelių savybių požiūriu: morfologiniu – gramatinių reikšmių reiškimo laipsnis (analitinė ar sintetinė), žodžių tvarka sakinyje, žodžių darybos būdas, abėcėlės ženklai, skiemenavimas (žodžių kėlimo taisyklės). Taip pat nurodysime ta kalba kalbančių žmonių apytikslį skaičių.

1 lentelė. Germanų kalbų grupė

	Anglų	Danų	Islandų	Norvegų	Olandų	Švedų	Vokiečių
Morfologija	iš esmės analitinė	iš esmės analitinė	sintetinė, tačiau naudoja prielinks-nius	iš esmės analitinė	iš esmės analitinė	iš esmės analitinė	iš esmės analitinė
Žodžių tvarka sakinyje	veiksnys-tarinys-papildinys-aplinkybės	veiksnys-tarinys-papildinys-aplinkybės	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	veiksnys-tarinys-papildinys-aplinkybės	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	veiksnys-tarinys-papildinys-aplinkybės	veiksnys-tarinys-papildinys-aplinkybės; vartojama ir kitokia tvarka
Žodžių daryba	sufiksacija, mažai kompozicinė	kompozicinė	kompozicinė	kompozicinė	kompozicinė	kompozicinė	kompozicinė
Raštas	lotynų	lotynų	lotynų	lotynų	lotynų	lotynų	lotynų
Skiemenavimas	panašu į lietuvių (galima palikti ir vieną raidę)	panašu į lietuvių	keliama pagal prasmę, skambesį, kitaip keliama po priebalsės	panašu į lietuvių, tik negalima kelti ar palikti vienos raidės	panašu į lietuvių	panašu į lietuvių, tik negalima kelti ar palikti vienos raidės	panašu į lietuvių
Kalbantieji	Jungtinė Karalystė – apie 50 mln., Airija – 3,5 mln.	Danija – 5,5 mln.	Islandija – 0,3 mln.	Norvegija 4,6 mln.	Nyderlandai – 16 mln., Belgija – 6 mln.	Švedija – 8,5 mln., Suomija – 0,3 mln.	Vokietija – 78 mln., Austrija – 7,7 mln., Šveicarija – 4,8 mln., Belgija – 0,1 mln., Rumunija – 0,3 mln., Liuksemburgas.

2 lentelė. Romanų kalbų grupė

	Ispanų	Italų	Katalonų	Portugalų	Prancūzų	Rumunų
Morfologija	beveik sintetinė	sintetinė, daug veiksmažodžių formų, bet nėra linksnių	sintetinė, tačiau turi analitinių bruožų, santykiai tarp daiktavardžių išreiškiami prielinksniais	sintetinė, daug veiksmažodžių formų, santykiai tarp daiktavardžių išreiškiami prielinksniais	iš esmės analitinė	iš esmės sintetinė
Žodžių tvarka sakinyje	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	mišri	veiksnys-tarinys-papildinys, galima veiksmažodį iškelti į pradžią	veiksnys-tarinys-papildinys	veiksnys-tarinys-papildinys, galima veiksmažodį iškelti į pradžią
Žodžių daryba	menkai kompozicinė		menkai kompozicinė	menkai kompozicinė	menkai kompozicinė	labai menkai kompozicinė
Raštas	lotynų	lotynų	lotynų	lotynų	lotynų	lotynų
Skiemenavimas	nesiejama su prasme, keliama pagal formalias taisykles	nesiejama su prasme, keliama pagal formalias taisykles	nesiejama su prasme, keliama pagal formalias taisykles	nesiejama su prasme, keliama pagal formalias taisykles	panašu į lietuvių	visiškai kaip lietuvių
Kalbantieji	Ispanija – 46 mln., Meksika, Argentina, Kolumbija.	Italija – 60 mln., Šveicarija – 0,5 mln.	Ispanija – 6,6 mln.	Portugalija – 10 mln., Angola – 10 mln., Brazilija, Rytų Timūras, Mozambikas.	Prancūzija – 62 mln., Belgija – 4 mln., Šveicarija – 1,5 mln., Liuksemburgas, Kanada, Haitis.	Rumunija – 23 mln.

3 lentelė. Slavų kalbų grupė

	Bulgarų	**Čekų**	**Lenkų**	**Serbų**	**Slovakų**	**Slovėnų**	**Rusų**
Morfologija	veiksma-žodžių atžvilgiu – sintetinė ir sudėtingesnė negu kitų slavų kalbų, kitų kalbos dalių – analitinė	iš esmės sintetinė, daug daiktavar-džių ir veiksma-žodžių galūnių	sintetinė su gausia linksniuo-čių sistema	sintetinė su gausia linksniuočių sistema	iš esmės sintetinė	sintetinė, daug linksniuočių	sintetinė su gausia linksniuo-čių sistema
Žodžių tvarka	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	beveik laisva	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)	veiksnys-tarinys-papildinys-aplinkybės	laisva	veiksnys-tarinys-papildinys-aplinkybės (su išlygom)
Žodžių daryba			menkai kompozicinė				
Abėcėlė	kirilica	kirilica	lotynų	lotynų, kirilica	lotynų+5 raidės	lotynų+3 raidės	kirilica
Skiemena-vimas	nesiejama su prasme, keliama pagal formalias taisykles	panašu į lietuvių	panašu į lietuvių	panašu į lietuvių	kaip čekų	nesiejama su prasme, keliama pagal formalias taisykles	panašu į lietuvių
Kalbantieji	Bulgarija – 7,3 mln.	Čekija – 10 mln.	Lenkija – 38 mln.	Serbija – 9,4 mln.	Slovakija – 5,4 mln.	Slovėnija – 2 mln.	Rusija – 143 mln. ir daug kitų Rytų Europos šalių gyventojų

4 lentelė. Baltų kalbų grupė

	Lietuvių	Latvių
Morfologija	daugiausia sintetinė	daugiausia sintetinė
Žodžių tvarka	beveik laisva	beveik laisva
Žodžių daryba	kompozicinė	kompozicinė
Abėcėlė	lotynų	lotynų
Skiemenavimas		kaip lietuvių
Kalbantieji	Lietuva – 3 mln.	Latvija – 2 mln.

5 lentelė. Ugrų-finų kalbų grupė

	Estų	Suomių	Vengrų
Morfologija	labai sintetinė	labai sintetinė	smarkiai sintetinė
Žodžių tvarka	veiksnys-tarinys-papildinys, tačiau vis dažniau vartojama laisva tvarka	veiksnys-tarinys-papildinys, tačiau galima ir keisti	visi pažymimieji žodžiai eina prieš daiktavardį
Žodžių daryba		kompozicinė	
Abėcėlė	lotynų	lotynų	lotynų
Skiemenavimas		panašu į lietuvių	kaip lietuvių
Kalbantieji	Estija – 1 mln.	Suomija – 4,9 mln.	Vengrija – 9,6 mln.

2.3. Rašto ženklai

Svarbiausias dalykas lokalizacijoje yra kalboje visuotinai vartojamų rašto ženklų rinkinys –raidės, ideografiniai ženklai, skaitmenys, skyrybos ženklai ir kiti ženklai. Be jų lokalizacija neturėtų prasmės. Todėl programų autoriai įvairių kalbų ženklus turinčias koduotes įdeda į originalias, dar nelokalizuotas programas, ką galima pavadinti pirmuoju žingsniu internacionalizacijos ir lokalizacijos link.

Koduotėms svarbi ženklų aibė, o daugeliui programų – ir daugelis kitų ženklų savybių: ženklų rikiavimas, didžiųjų ir mažųjų raidžių atitiktis, raidžių variantai priklausomai nuo jų vietos žodyje, skyrybos ženklų įvairovė ir kt.

Abėcėlinėse (raidinėse ir skiemeninėse) kalbose ženklų būna kelios dešimtys, ideografinėse (kinų, japonų, korėjiečių) kalbose ženklų (hieroglifų) yra tūkstančiai. Todėl šių dviejų kalbų grupių rašto ženklų kodavimas smarkiai skiriasi.

Ideografinės kalbos pasižymi rašto ženklų gausa, o abėcėlinės – raštų gausa. Į unikodą įtraukti lotynų, kirilicos, graikų, arabų, armėnų, gruzinų, hebrajų, bengalų, kanadų, gudžaračių, devanagari, khmerų, etiopų, laosiečių ir kiti rašmenys.

Daugiausia kalbų vartoja lotynų rašmenis. Jų abėcėlės turi daug bendrų raidžių, tačiau kiekviena kalba turi ir jai savitų raidžių. Į unikodą įtraukta apie 350 lotynų raidžių. Dar nemažai jų galima gauti naudojant kompozicines ženklų sekas (žr. 3.4.2 skyr.).

Yra kalbų, kuriose vartojama daugiau negu vienas raštas. Pavyzdžiui, serbai vartoja lotynų arba kirilicos rašmenis, Lietuvoje gyvenantys karaimai vartoja lotynų rašmenis, o Rusijoje gyvenantys – kirilicą.

2.3.1. Lotynų abėcėlės ir kiti ženklai

Ženklų aibė. Visų kalbų, vartojančių lotyniškus rašmenis, abėcėlių pagrindas yra lotynų kalbos abėcėlė atmetus tai kalbai nereikalingas raides ir papildžius savitomis tai kalbai reikalingomis raidėmis, kurių nėra lotynų kalbos abėcėlėje.

Klasikinė lotynų kalbos abėcėlė turi 23 raides:

A B C D E F G H I K L M N O P Q R S T V X Y Z

Kompiuterijoje lotynų abėcėle vadinama klasikinė lotynų kalbos abėcėlė, papildyta raidėmis J, U ir W. Ji faktiškai sutampa su anglų kalbos abėcėle, irgi turinčia 26 raides. Kitų kalbų abėcėlėse dauguma naujų raidžių gaunama iš papildytos lotynų kalbos abėcėlės raidžių, prie jų pridėjus diakritinius ženklus (pvz., Ą, Č) arba kitaip modifikavus (pvz., Æ, Ł). Visos jos yra lotyniškos, t. y. priklauso lotynų rašmenims, tačiau ne visos priklauso lotynų kalbos abėcėlei. Atsiranda nevienareikšmiškumų dėl to, kad tuo pačiu pažymimuoju žodžiu „lotynų“ apibūdinama ir lotynų kalba, ir lotynų raštas. Siekiant tikslumo, *lotynų kalbos abėcėle* vadinsime klasikinės lotynų kalbos 23 raidžių abėcėlę, o iki 26 raidžių išplėstą lotynų kalbos abėcėlę – *pagrindine lotynų abėcėle* (be žodžio „kalbos“) arba *anglų kalbos abėcėle*.

Lietuvių kalboje vartojami lotynų rašmenys, kuriems priklauso visos 32 lietuvių kalbos raidės. Iš jų 9 raidės (Ą, Č, ...) nepriklauso nei pagrindinei lotynų abėcėlei, nei lotynų kalbos abėcėlei. Jas vadinsime savitosiomis lietuvių kalbos raidėmis.

Lietuvių kalbos garsams išreikšti abėcėlėje yra 30 raidžių. Kitos dvi – F ir H – perimtos iš svetur, kartu su jomis išreiškiamais garsais. Jos yra visateisės lietuvių kalbos abėcėlės raidės.

Septyniolika lietuvių kalbos abėcėlės raidžių (A, Ą, E, Ę, Ė, I, Į, Y, J, L, M, N, O, R, U, Ų, Ū) gali būti kirčiuotos. Kirčiuotos raidės į abėcėlę neįtrauktos, vartojamos tik tam tikruose tekstuose, pavyzdžiui, žodynuose, vadovėliuose. Gali, bet neprivalo, būti vartojamos ir visuose kituose tekstuose.

Dar specifiškesnę raidžių grupę sudaro transkripcijos ženklai, senųjų raštų raidės. Jų yra gana daug, specialiai tokiems rašmenims parengtas „Palemono“ šriftas (Aleknavičienė, Grumadienė ir kt., 2005).

Taigi raidžių aibė, jų vartojimas ir klasifikacija lietuvių kalboje aiškiai apibrėžti. Abėcėlė – taip pat – 32 raidės.

Kaip atrodo lietuvių kalbos abėcėlė tarp kitų Europos kalbų, vartojančių lotyniškus rašmenis ir esančių pagrindinėmis oficialiomis (valstybinėmis) kalbomis, o taip pat lotynų ir esperanto abėcėlių, matyti 6-oje lentelėje.

Lentelėje išvardijamos ir skaičiuojamos tik didžiosios raidės. Taigi faktiškai raidžių yra dvigubai daugiau.

Kai kuriose kalbose yra pasenusių, bet dar tebevartojamų ar šiaip retai vartojamų raidžių[1]. Yra raidžių, vartojamų tik svetimžodžiuose, tikriniuose svetimų kalbų varduose, tik adaptuotuose arba tik neadaptuotuose varduose ir pan. Dėl to kai kurios tokios raidės skirtingai traktuojamos skirtinguose šaltiniuose, pateikiančiuose tos pačios kalbos abėcėlę. Sudarydami šią lentelę daugiausia rėmėmis informacija, pateikta standartuose ISO 12199 ir ETSI ES 202 130, o ten, kur duomenys skiriasi, užpildydami lentelę vadovavomės panašiais kriterijais, kuriais apibrėžiamos lietuvių kalbos abėcėlės raidės.

Aptarsime išskirtinesnius atvejus.

Istoriškai latvių kalbos abėcėlėje buvo vartojamos raidės Ō ir Ŗ. Po 1946 metais įvykdytos kalbos reformos buvo atsisakyta raidės Ŗ, o vėliau ir Ō. Tačiau kai kurios bendrijos, ypač išeivijoje, jas vartoja iki šiol. Jas vartoja savaitraštis „Brīvā Latvija“ (Laisvoji Latvija), jos įtrauktos į koduotes.

Raidė Å į danų kalbos abėcėlę buvo įtraukta 1948 metais.

Raidė Z iš islandų kalbos abėcėlės išbraukta 1974 metais, bet dar išliko vienos vidurinės mokyklos „Verzlunarskóli Íslands“ pavadinime, ją retkarčiais vartoja populiarus laikraštis „Morgunblaðið“ (Ryto laikraštis).

Olandų kalboje vartojama keletas raidžių su diakritiniais ženklais, bet jų statusas yra panašus į kirčiuotų raidžių statusą lietuvių kalboje.

[1] Jeigu į lietuvių kalbos abėcėlę priimtume ir kirčiuotas raides, tai ir mes turėtume vieną labai retą raidę – kirčiuotą J su riestiniu kirčiu (pvz., žodyje danguĵ).

6 lentelė. Įvairių Europos kalbų abėcėlės

Kalba	Kalbos kodas	Neturi pagrindinių lotynų raidžių	Turi savitųjų raidžių	Raidžių skaičius abėcėlėje		
		a	*b*	*a*	*b*	26−*a*+*b*
Airijos gėlų[1]	ga	J K Q W X Y Z	Á É Í Ó Ú	7	5	24
Albanų	sq	W	Ç Ë	1	2	27
Anglų	en			0	0	26
Čekų	cz	Q W X	Á Č Ď É Ě Í Ň Ó Ř Š Ť Ú Ů Ý Ž	3	15	38
Danų	da		Æ Ø Å	0	3	29
Estų	et	C Q W X Y	Ä Ö Õ Ü	5	4	25
Islandų	is	C Q W Z	Á Đ É Í Ó Ú Ý Þ Æ Ö	4	10	32
Ispanų	es		Á É Í Ñ Ó Ú Ü	0	7	33
Italų	it	J K X W Y	À É È Ì Ò Ó Ù	5	7	28
Lenkų	pl	Q V X	Ą Ć Ę Ł Ń Ó Ś Ź Ż	3	9	32
Latvių	lv	Y Q W X	Ā Č Ē Ģ Ī Ķ Ļ Ņ Š Ū Ž	4	11	33
Lietuvių	lt	Q W X	Ą Č Ę Ė Į Š Ų Ū Ž	3	9	32
Lotynų	la	J U W		3	0	23
Maltiečių	mt	C Y	Ċ Ġ Ħ Ż	2	4	28
Norvegų	no		Æ Ø Å	0	3	29
Olandų	nl			0	0	26
Portugalų	pt		Á À Â Ã Ç É Ê Í Ó Ô Õ Ú	0	12	38
Prancūzų	fr		À Â Ç Œ Ô Ù Û Ü Ÿ È É Ê Ë Î Ï	0	15	41
Rumunų	ro		Â Ă Î Ş Ţ	0	5	31
Slovakų	sk	Q W X	Á Ä Č Ď É Í Ĺ Ľ Ň Ó Ŕ Š Ť Ú Ý Ž	3	16	39
Slovėnų	sl	Q W X Y	Č Š Ž	4	3	25
Suomių	fi	B C F Q W	Å Ä Ö	5	3	24
Švedų	se		Å Ä Ö	0	3	29
Turkų	tr	Q W X	Ç Ğ İ Ö Ş Ü	3	6	29
Vengrų	hu	Q W X Y	Á É Í Ó Ö Ő Ú Ü Ű	4	9	31
Vokiečių	de		Ä Ö ß Ü	0	4	30

[1] Gėlų kalbą vartoja airiai, škotai ir kitos tautos. Airijoje vartojamas gėlų kalbos variantas dažnai vadinamas airių kalba.

Brazilijos portugalų kalboje iki 2008 metų buvo vartojama raidė Ü. Nuo 2009 metų ją leidžiama vartoti tik skoliniuose, asmenvardžiuose ir iš jų padarytuose žodžiuose. Tai daugelį metų rengtos portugalų kalbos reformos, siekiančios suvienodinti įvairių kalbos atmainų, vartojamų įvairiose valstybėse, rašybą, rezultatas.

Tikros rumunų abėcėlės raidės yra Ș ir Ț (S su kableliu ir T su kableliu). Jos ilgą laiką nebuvo įtrauktos į koduotes ir šriftus. Vietoj jų buvo vartojamos į jas panašios Ş ir Ţ (S su sedile ir T su sedile). Dabar galimybė vartoti šias raides jau yra, bet įprotis vartoti pakaitalus išliko. Todėl rumunų kalboje tebėra painiavos su perkodavimais.

Vokiečių kalboje nėra žodžių, prasidedančių raide ß. Ši raidė buvo tik mažoji, o 2007 metais į unikodą įtraukta ir didžioji.

Kaip matome, abėcėlių įvairovė gana didelė, lietuvių kalba savitosiomis savybėmis neišsiskiria.

Lentelėje pateiktų kalbų raidžių skaičius abėcėlėje svyruoja nuo 23 (Airijos gėlų, klasikinė lotynų) iki 41 (prancūzų). Vidutiniškai abėcėlė turi 30 raidžių.

Daugelyje kalbų nevartojamos kai kurios pagrindinės lotynų abėcėlės raidės, dažniausiai tos pačios, kaip ir lietuvių kalboje: W (14 kalbų iš 26), Q (12), X (11).

Didžiosios ir mažosios raidės. Būna raidžių, kuriomis neprasideda joks kalbos žodis: čekų ir lenkų Ą, Ę, Ń, lietuvių Ę, Ų. Tokių didžiųjų raidžių prireikia retai, tik užrašant visą žodį vien didžiosiomis raidėmis. Kai kurių kalbų (čekų, lenkų) klaviatūrose jos „nustumiamos“ į rinkimui mažiau patogias vietas, pavyzdžiui, jos išgaunamos su kombinaciniu klavišu, o jų tradicinė vieta (mažosios raidės klavišo antrajame lygyje) užleidžiama reikalingesniems ženklams.

Didžiųjų raidžių vartojimas taip pat skiriasi. Vokiečiai didžiąja raide pradeda visus daiktavardžius, JAV anglų kalbos vartotojai – visus arba beveik visus žodžius antraštėse.

Yra raštų (kalbų), iš viso nevartojančių didžiųjų raidžių (pvz., arabų, gruzinų, hebrajų).

Raidės su diakritiniais ženklais. Tas pats diakritinis ženklas, uždėtas ant raidės (gali būti ir greta jos arba po ja), priklausomai nuo kalbos, gali atlikti dvejopą funkciją: 1) sukurti naują raidę, kuri įtraukiama į kalbos abėcėlę arba 2) suteikti papildomos informacijos apie raidę, ant kurios jis uždėtas (tokios raidės į abėcėlę dažniausiai neįtraukiamos).

Raidės Ą, Č, Ę, Ė, Į, Š, Ų, Ū, Ž yra įtrauktos į lietuvių kalbos abėcėlę. Todėl jos yra privalomos. Kirčio ženklai naujų raidžių nesukuria. Todėl juos ant raidžių galima rašyti, galima ir nerašyti. Jie neprivalomi.

Ta pati Õ estams yra atskira abėcėlės raidė, o mums – tik raidė O su riestiniu kirčiu.

Dešiniojo kirčio ženklą danai kartais uždeda ant balsių, kai nuo tarimo (kirčio) priklauso žodžio reikšmė, pavyzdžiui, *jeg stód op* (aš stoviu), *jeg stod óp* (aš keliuosi).

Dvejopą diakritinio ženklo paskirtį mes gerai suvokiame. Kitų kalbų, ypač tų, kurių abėcėlėse nėra raidžių su diakritiniais ženklais, atstovai ją skiria sunkiau. Anglų kalboje nėra ir kirčiuotos raidės sąvokos: ir kirčiuota, ir raidė su diakritiniu ženklu vadinama vienodai: *accented letter*. Dėl to pasitaiko internacionalizacijos klaidų, kai raidės su diakritiniais ženklais ignoruojamos.

Taškas ant i. Taškas, kaip ir bet koks kitas ženklelis, uždėtas ant raidės, laikomas diakritiniu ženklu. Pavyzdžiui, jį uždėję ant Ee, gauname Ėė (lietuvių k.), ant Ze – Żż (lenkų k.). Bet nesakome, „i su tašku“, nes mažoji i lietuvių ir daugelyje kitų kalbų visada su tašku. Be to, ant didžiosios I taško nededame. Taigi, taškas ant i raidės nelaikomas diakritiniu ženklu. Bet turkų ir azerbaidžanų abėcėlėse yra ir raidė i be taško: ı (U+131), ji taip ir vadinama: „i be taško“. Turi jie ir mūsiškę Ii raidę. Tačiau mūsiškė didžioji I ir taip be taško! Tam, kad turkų kalbos abėcėlėje raidės i ir ı turėtų atskiras didžiąsias, teko pripažinti didžiąją I su tašku (U+0130). Taigi turime:

I i
I ı – i be taško
İ i – i su tašku (turkų k.)

Ir dar: daugelio kalbų abėcėlėse uždėjus ant i kokį nors diakritinį ženklą gaunamos naujos raidės, pavyzdžiui, Īī, Íí , Ǐǐ, Ȉȉ, Îî, Ǐǐ, tada taškas nededamas. Tačiau ant lietuviškos kirčiuotos i taškas išlieka, pavyzdžiui, í. Tai galima paaiškinti tuo, kad kirčiuota raidė nėra nauja raidė, o tik nedidelė pamatinės raidės modifikacija, todėl nedera keisti pamatinės raidės piešinio, prie jo galima tik šį tą pridėti.

Priklausomybė nuo vietos žodyje. Vokiečių kalboje keliant žodį į kitą eilutę ties raide ß, ši raidė skyla į dvi raides: *Straße –> Stras-se*. O kai perkeltos žodžio dalys sujungiamos, ss vėl virsta ß.

Graikų kalboje raidė σ (sigma), atsidūrusi žodžio pabaigoje, keičiama kitokio pavidalo raide ς (sigma), pavyzdžiui, ίσος.

Kiti ženklai

Be raidžių tekste vartojami skaitmenys, skyrybos ženklai, matematikos ir kiti ženklai. Trumpai aptarsime tuos, kurie labiausiai skiriasi įvairiose kalbose.

Kabutės. Tai didžiausią įvairovę turintis skyrybos ženklas.

„...“ bulgarų, čekų, estų, islandų, lietuvių, kroatų, serbų, slovakų, sorbių, vokiečių;

“...” airių, anglų, hebrajų;

„...” afrikanų, lenkų, olandų, rumunų, vengrų;
”...” suomių, švedų;
«...» latvių, rusų, albanų, baltarusių, graikų, ispanų, katalonų, norvegų, portugalų, turkų, ukrainiečių;
« ... » prancūzų, tarp kabučių ir jomis gaubiančio yra tarpas;
»...« danų;
〝文字〃 ,
「文字」 kinų, arba kai rašoma iš viršaus į apačią:

┐
文
字
└

Kai kuriose kalbose dar vartojamos antrojo lygio kabutės – kabutės jau esančiame tarp kabučių tekste, pavyzdžiui, anglų kalboje jos yra tokios: ‘...’.

Klaustukas ir šauktukas. Ispanų kalba turi du klaustukus: ¿ ir ?, ir du šauktukus ¡ ir !. Vienas (mūsų požiūriu – apverstas) rašomas sakinio pradžioje, kitas – pabaigoje, pavyzdžiui:

¿Qué hora es?
¡Cuidado!

Logiška: jau pradedant skaityti sakinį matosi, kokia intonacija jį perskaityti.

Graikų kalboje klaustuko funkciją atlieka ženklas ; (U+307E), kurio išvaizda nesiskiria nuo mūsiškio kabliataškio, o mūsiškio kabliataškio funkciją atlieka taškas viršuje · (U+0387).

Numerio ženklas. Kai kurios kalbos turi specialų numerio ženklą:

anglų k. (JAV)
№ rusų k.

2.3.2. Rikiavimas

Vienas iš teksto (žodžių, eilučių) rikiavimo būdų yra rikiavimas pagal abėcėlę. Raidžių išdėstymo eilė abėcėlėje lemia ir jų rikiavimą. Tačiau yra ir tam tikrų niuansų: ar didžiąsias ir mažąsias raides laikyti lygiavertėmis, ar skirti raidę nuo jos modifikacijos (pvz., A nuo Ą), kaip įvertinti skyrybos ir kitus tekste pasitaikančius ženklus, kurių nėra abėcėlėje. Todėl įprasta rikiuoti palaipsniui, pagal lygius, kaskart patikslinant. Atskirai panagrinėsime rikiavimą lietuvių kalboje, kitose kalbose ir daugiakalbį rikiavimą.

Rikiavimas lietuvių kalboje.

1 lygis. Rikiavimas pagal raidžių grupes, neskiriant didžiųjų raidžių nuo mažųjų:

AĄ B C Č D EĘĖ F G H IĮY J K L M N O P R S Š T UŲŪ V Z Ž

Lygiavertėmis laikomos didžiosios ir jas atitinkančios mažosios raidės, raidžių grupės, į kurią patenka pagrindinė raidė ir jos modifikacijos. Lygiaverčių raidžių grupes skyrėme tarpais. Be mažųjų ir didžiųjų raidžių lygiavertėmis laikomos dar šios raidžių grupės:

AĄ;
EĘĖ;
IĮY;
UŲŪ.

Atkreipiame dėmesį, kad C ≠ Č, S ≠ Š ir Z ≠ Ž, t. y. šios raidės nelygiavertės.

Jeigu surikiavus pirmu lygiu dar lieka rikiavimo požiūriu lygiaverčių sąrašo elementų, rikiuojama antru lygiu. Rikiuojant pagal aukštesnį lygį iš naujo perrikiuojami tie sąrašo fragmentai, kurie žemesniame lygyje buvo lygiaverčiai.

2 lygis. Rikiavimas pagal atskiras raides, neskiriant didžiųjų raidžių nuo mažųjų:

A Ą B C Č D E Ę Ė F G H I Į Y J K L M N O P R S Š T U Ų Ū V Z Ž

Lygiavertėmis laikomos tik didžiosios ir jas atitinkančios mažosios raidės.

Nosinė raidė eina tuojau pat po raidės, iš kurios ji padaryta: EĘĖ, UŲŪ. Taip logiškiau, nes nosinė raidė yra tik istoriškai atsiradusi raidės modifikacija, o raidžių Ė ir Ū žymimi garsai skiriasi nuo jų pagrindinių raidžių garsų.

3 lygis. Nėra lygiaverčių raidžių – visos laikomos skirtingomis:

A a Ą ą B b C c Čc D d E e Ę ę Ė ė F f G g H h I i Į į Y y J j K k L l M m N n O o P p R r S s Š š T t U u Ų ų Ū u V v Z z Ž ž

Didžioji raidė laikoma pirmesne už mažąją.

Jeigu tekste yra kitų kalbų abėcėlių raidžių, tai tada raides Q, W, X ir Þ, kurios yra daugiakalbio rikiavimo standarto ISO 12199:2000 pirmajame lygyje, reikėtų laikyti keturiomis pirmojo lygio grupėmis ir įtraukti į bendrą eilę, į vietas, atitinkančias minėto standarto rikiavimo eilę PQR, VWXZÞ.

Rikiavimas kitose kalbose. Bendri rikiavimo principai – skirstymas į lygius, skaitmenų rikiavimas – iš esmės yra tokie pat ir kitose kalbose. Tačiau skiriasi abėcėlės ir atskirų jos raidžių vertinimas. Todėl vienos kalbos rikiavimo taisyklės beveik visada netinka kitai kalbai. Ypač daug įvairovės atsiranda dėl dviraidžių ir triraidžių. Tai dviem arba trim raidėmis užrašytas garsas. Jie turi atskiras vietas abėcėlėje ir kai

žodžiai rikiuojami, dviraidžius ir triraidžius sudarančios dvi arba trys raidės laikomos vienu simboliu (ženklu), išliekančiu nedalomu visuose rikiavimo lygiuose. Todėl, pavyzdžiui, čekų kalbos žodžiai *chalupa, cvok* ir *husa* rikiuojami taip:

cvok,
husa,
chalupa,

nes ši kalba turi dviraidį CH. Be to, dviraidis CH čekų abėcėlėje eina po raidės H, todėl žodis *chalupa* atsidūrė po žodžio *husa*.

Anksčiau dviraidį CH turėjo ir lietuvių kalba. Dar ir dabar pasitaiko žodynų, į kurių pradžią įdėtose abėcėlėse CH pateikiama kaip atskira raidė, bet išdėstant žodžius į šį dviraidį nebeatsižvelgiama.

ISO 12199:2000 standarto informaciniame priede pateikta informacija apie 14 kalbų (iš 50) dviraidžius ir triraidžius:

albanų: dh gj ll nj rr sh th xk zh;
baskų: ll tt;
čekų: ch;
fryzų: ij;
ispanų (ch ll);
katalonų: ll;
korsikiečių: ghj chj;
kroatų: dž lj nj;
maltiečių: gh;
slovakų: ch;
somaliečių: dh kh sh;
svahilių: ch dh gh ng' ny sh th;
velsiečių: ch dd ff ng ll ph rh th;
vengrų: cs dz dzs gy ly ny sz ty zs.

Ispanų kalbos dviraidžius rašėme į skliaustus dėl to, kad 2004 metais ispanai rikiuodami juos nusprendė išskirti į atskiras raides, bet abėcėlėje palikti dviraidžiais.

Įdomią rikiavimo taisyklę turi prancūzų kalba. Žodžiuose, kurie skiriasi tik raidėmis su diakritiniais ženklais, raidės lyginamos peržiūrint žodį iš dešinės į kairę. Todėl, pavyzdžiui, žodžiai *cote*, *coté* ir *côte* rikiuojami taip:

cote,
côte,
coté.

Daugiakalbis rikiavimas. Daugiakalbio teksto rikiavimo taisykles apibrėžia ISO 12199 standartas. Pradžioje pateiksime lotynų abėcėlės raidžių rikiavimo taisykles.

1 lygis. Rikiavimas pagal raidžių grupes, neskiriant didžiųjų raidžių nuo mažųjų.

Grupių pagrindu imamos 27 lotynų abėcėlės raidės:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Þ (*)

Prie šių 27 raidžių dar prijungiamos dvi raidžių grupės.

1 grupė. Kitos lotynų abėcėlės raidės, išskyrus raides su diakritiniais ženklais, pakeistos į jas panašiomis raidėmis arba jų poromis:

Æ → AE; (**)
Ɓ → B;
Ƈ → C;
Ð → D;
Ɗ → D;
Đ → D;
Ɠ → G;
Ħ → H;
ĸ → k;
Ł → L;
Ŋ → N;
Ø → O;
Œ → OE;
ß → ss;
Ŧ → T.

Trys raidės (Æ, Œ ir ß) skeliamos į dvi raides.

Išvardijome tik didžiąsias, bet ta pati taisyklė taikoma ir mažosioms raidėms. Dvi raidės didžiųjų neturėjo, todėl rašėme tik mažąsias. Taip buvo standarte. Dabar viena jų – ß turi ir didžiąją, kuri keičiama į SS.

2 grupė. Raidės su diakritiniais ženklais prijungiamos prie tų raidžių, iš kurių jos gautos pridėjus diakritinius ženklus.

2 lygis. Rikiavimas pagal atskiras raides neskiriant didžiųjų raidžių nuo mažųjų.

Pirmojo lygio raidžių grupės išskirstomos į raidžių poras (mažoji, didžioji) ir kiekvienos grupės poros surikiuojamos tokia eile:

1. Pagrindinė pora, nurodyta pirmojo lygio pagrindinių porų sąraše (*).

2. Raidžių poros (jei yra), įtrauktos į pirmojo lygio grupes pagal (*) sąrašą ir surikiuotos pagal (**) sąrašą.
3. Raidžių poros (jei yra) su diakritiniais ženklais, surikiuotos pagal diakritinius ženklus, kurių rikiavimo eilė apibrėžta ISO 12199 standarte. Toje eilėje pirmiau išdėstyti ženklai, rašomi virš raidės, po jų ženklai, rašomi žemiau raidės.

3 lygis. Nėra lygiaverčių raidžių – visos laikomos skirtingomis. Didžioji raidė eina po ją atitinkančios mažosios.

Daugiakalbiame rikiavime raidžių vietą surikiuotoje eilėje lemia raidžių forma: turi ar neturi diakritinį ženklą, ar diakritinis ženklas virš raidės, ar po ja. Lietuvių kalboje, o taip pat ir daugelyje kitų kalbų, pirmenybė teikiama ženklų semantikai. Tų pačių ženklų forma vienoda visose kalbose, o semantika skiriasi. Įvertinti semantiką daugiakalbiame rikiavime būtų sunku – kas tinka vienai kalbai, dažnai netinka kitai. Todėl formos pasirinkimą daugiakalbiame rikiavime reikėtų laikyti racionaliu sprendimu, nors jis labai formalus ir skiriasi nuo rikiavimo daugelyje atskirų kalbų.

ISO 12199 standarto informaciniame priede pateikta informacija rodo, jog jau po pirmojo lygio rikiavimo iš 60 kalbų sąrašo liktų 15 kalbų, kurių rikiavimo rezultatas sutaptų su daugiakalbio rikiavimo rezultatu (tame 15 kalbų sąraše tik 4 yra oficialios Europos Sąjungos kalbos: airių, italų, portugalų ir prancūzų). Po trečiojo lygio sutampančių sąrašas sutrumpėtų iki 13 (jame liktų 3 Europos Sąjungos kalbos: airių, italų ir portugalų).

Lietuvių kalbos abėcėlės rikiavimas nuo tarptautinio skiriasi tuo, kad:

1. Raidė Y prilyginama raidei I, o raidės Č, Š ir Ž neprilyginamos jokioms kitoms raidėms (laikomos visiškai nepriklausomomis). Tai atitinka raidžių semantiką: raidė Y žymi tą patį garsą, kaip ir I, tik ilgesnį. Raidės Č, Š ir Ž žymi kitus garsus, negu C, S ir Z.
2. Nosinės raidės eina tuojau pat po pagrindinės raidės aplenkdamos tokias pat raides su diakritiniais ženklais virš jų. Jos žymi tokius pat garsus, kaip ir jas atitinkančios pagrindinės raidės. Todėl natūralu, kad būtų arčiau pagrindinių, negu žyminčios tokius pat, tik ilgesnius garsus (Y, Ū) arba kitaip panašius (Ė).
3. Didžioji raidė eina prieš mažąją. Tai natūralu iš kalbos pozicijų: didžioji būna tik žodžio pradžioje. Daugiakalbiame rikiavime mažoji eina prieš didžiąją – čia panašiau į matematiką: didesnis skaičius eina po mažesnio, didžioji raidė – po mažosios.

Panašiai rikiavimas skiriasi ir kitose kalbose. Paminėsime keletą įdomesnių atvejų. Lenkai raidę Ł laiko atskira raide ir aukštesnio rango, negu raidės su diakritiniais ženklais. Estai raidę Z atkelia po raidės S.

Kitų raštų raidės rikiuojamos pagal tas pačias taisykles, tik kiekvienas raštas turi savą raidžių rikiavimo eilę. Graikų raidės rikiuojamos taip:

Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω

Tai pagrindinės graikų abėcėlės raidės. Be jų specialiuose tekstuose vartojamos toninės raidės. Tai panašu į kirčiuotas lietuvių raides.

Visos kirilicos raidės yra pirmojo lygio, t. y. išdėstomos į viena eilę jau pirmuoju rikiavimo lygiu. Pateiksime dažniau vartojamų kirilicos raidžių išdėstymą:

А Б В Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ю Я Ъ Ь Ђ Љ Њ Ћ Џ

Daugiakalbiame tekste gali būti įvairių raštų raidžių. Į bendrą eilę skirtingų raštų raidės sudedamos tokia tvarka:

lotynų;
graikų;
kirilica;
kiti raštai.

Skirtingų raštų raidės atsiskiria jau pirmajame rikiavimo lygyje. Minėtų trijų raštų raidžių eilutė (po penkias didžiąsias raides iš kiekvieno) būtų surikiuota taip:

D E K L M Δ Ε Κ Λ Μ Д Е К Л М

Skaitmenų rikiavimas. Skaitmenys rikiuojami pagal jų vertes:

0 1 2 3 4 5 6 7 8 9

Tekste skaitmenys laikomi teksto ženklais ir rikiuojami kaip atskirų ženklų sekos, o ne kaip iš jų sudaryti skaičiai. Surikiuotų skaitmenų eilučių pavyzdys:

1
10
100
11
2
21

Tam, kad skaitmenų eilutės būtų surikiuotos pagal jomis išreikštų skaičių vertes, reikia suvienodinti jų ilgius įterpiant į jų pradžias nulių:

001
002

010
011
021
100

Skaitmenys eina prieš raides.

Skaitmenų semantika visose kalbose vienoda, nors jų forma gali ir skirtis. Jų rikiavimas problemų nekelia.

Specialiųjų ženklų rikiavimas. Paprastai specialieji ženklai rikiuojami pirmiau skaitmenų ir raidžių.

2.3.3. Kirčiuotos raidės ir lituanistikos ženklai

Kirčiuotos raidės. Tai išskirtinė lietuvių kalbos savybė. Lietuvių kalba turi 68 kirčiuotas raides: 34 didžiąsias ir tiek pat mažųjų.

Àà Áá Ãã	L̃l̃
Ą́ą́ Ą̃ą̃	M̃m̃
Èè Éé Ẽẽ	Ññ
Ę́ę́ Ę̃ę̃	Òò Óó Õõ
Ėė Ẽẽ	Ř ř
Ìì Íí Ĩĩ	Ùù Úú Ũũ
Į́į́ Į̃ į̃	Ų́ų́ Ų̃ų̃
Ýý Ỹỹ	Ű́ű̃ Ũ̃ũ̃
J̃j̃	

Kirčiuotos raidės ir jų kodavimas tarptautiniu lygiu pirmą kartą buvo pateiktas straipsnyje (Tumasonis, 1999). Apie kirčiuotų raidžių kodavimą rašoma 3.4.3 skyrelyje.

Lituanistikos ženklai. Tokių ženklų poreikis aiškiai suformuluotas monografijoje (Aleknavičienė ir kt., 2005, 13 p.): „Filologijai reikia daug daugiau ženklų nei kitiems mokslams: seniesiems raštams publikuoti originalia autoriaus rašyba, tarmių tekstams transkribuoti, indoeuropeistų bei baltistų rekonstruotoms formoms fiksuoti. Vien seniesiems raštams reikia kelių tos pačios raidės variantų – ta pati fonema net ir to paties autoriaus tekste dažnai realizuojama keletu alografų“.

Į „Palemono“ ženklų aibę ir į tuo pačiu vardu pavadintą šriftą įtraukta apie 4300 specialiųjų ženklų. Dalis jų reikalingi ir kitoms kalboms ir buvo įtraukti į unikodą

tų kalbų atstovų siūlymu, kiti koduojami kompozicinėmis sekomis arba įtraukti į privačiąją sritį. Lietuvių iniciatyva į unikodą įtrauktas kirstinės priegaidės *zigzagas* ◌̋, vartojamas lietuvių tarmių fonetinėje transkripcijoje (Aleknavičienė ir kt., 2004). Jo kodas: U+035B. Kiek vėliau – dar du ženklai: *kairysis samtis* (U+1DCB) ◌᷋ ir *dešinysis samtis* (U+1DCC) ◌᷌. Juos vartojo Antanas Baranauskas.

2.4. Ženklų identifikavimas

Rašto ženklus atpažįstame pagal jų piešinius. Tačiau ne visada vienareikšmiškai galima nustatyti, kurį ženklą atitinka piešinys. Pavyzdžiui, *Arial* šrifte visiškai sutampa mažosios raidės *l* ir didžiosios raidės *i* piešiniai:

l I

Times New Roman šrifte labai mažai skiriasi vienetas nuo mažosios raidės *l*:

1 l

Nulis daugelyje šriftų vaizduojamas siauresniu piešiniu, negu didžioji raidė O, bet kai kuriuose šriftuose, pavyzdžiui, *Algerian* jų piešiniai labai panašūs:

0 O

Daugeliu atvejų ženklą, pavaizduotą nevienareikšmiu piešiniu, galima nustatyti iš konteksto. Todėl ikikompiuteriniais laikais, kai tekstas buvo skiriamas tik žmogui, rašomosiose mašinėlėse nebuvo nulio ir vieneto. Vietoj jų spausdindavo didžiąją O ir mažąją L.

Kompiuteryje supainioti skaitmenį su raide būtų didelė klaida. Todėl imtasi juos skirti. Ypač svarbu buvo jų nesupainioti kompiuteriui ruošiamuose duomenyse ir programose. Todėl tais laikais, kai kompiuteriui pateikiami tekstai pirmiausiai buvo rašomi ranka į specialius blankus, buvo laikomasi tam tikrų susitarimų dėl nulio ir raidės O rašymo. Lietuvoje buvo priimta nulį perbraukti: Ø, o raidę O rašyti su uodegėle. Ši taisyklė tiko lietuviškiems rašmenims, bet, pavyzdžiui, norvegai turi O ir Ø raides. Jiems reikėjo kitokių susitarimų.

Kita problema – sutampantys kai kurių skirtingų raštų raidžių piešiniai. Pavyzdžiui, kirilicos didžiosios raidės P (rusiškos er) piešinys sutampa su lotyniškos didžiosios raidės P (pė) piešiniu. Sutampa lotyniškos A didžiosios, graikų A didžiosios (alfa didžiosios) ir kirilicos A didžiosios raidžių piešiniai.

Vienareikšmiškumas ypatingai svarbus koduočių standartų aprašuose. Todėl čia ženklų piešiniai atlieka tik informacinę (ne norminę) iliustracijos funkciją.

Siekiant vienareikšmiškumo buvo sutarta ženklus identifikuoti pavadinimais, pagal tam tikras taisykles įvardijant ženklų priklausomybę raštui (*lotynų, graikų* ir kt.), ženklų rūšį (*raidė, skaitmuo* ir pan.) ir kitus būdingus ženklų požymius. Pirmiausia ženklų pavadinimų sistema buvo sudaryta anglų kalba. Tam, kad kuo mažiau raidžių (ir kitų ženklų) būtų vartojama jų pačių pavadinimuose, apsiribota tik 26 didžiosiomis anglų kalbos abėcėlės raidėmis, tarpo ženklu ir brūkšneliu. Pavyzdžiui, minėtos trys didžiosios A, pavaizduotos tokiu pat piešiniu, įvardijamos taip:

A LATIN CAPITAL LETTER A,
A GREEK CAPITAL LETTER ALPHA,
A CYRILLIC CAPITAL LETTER A.

Mažosios raidės dėl sutarto ženklų aibės ribojimo žymimos didžiosios raidės glifu, bet į raidės pavadinimą įtraukiamas žodis SMALL, pavyzdžiui:

a LATIN SMALL LETTER A,
þ LATIN SMALL LETTER THORN.

Sudėtinės raidės įvardijamos pamatinės raidės ir jos priedų pavadinimais, pavyzdžiui,

ǻ LATIN SMALL LETTER A WITH RING ABOVE AND ACCUTE,
š LATIN SMALL LETTER S WITH CARON.

Pažodžiui išvertę šiuos pavadinimus į lietuvių kalbą turėtume:

ǻ lotynų mažoji raidė A su žiedu viršuje ir dešininiu kirčiu,
š lotynų mažoji raidė S su paukščiuku.

Kadangi raidę Š turime savo abėcėlėje, tai jos pavadinimo vertimas turėtų būti:

š lotynų mažoji raidė Š

Čia žodis *lotynų* (angl. *Latin*) reiškia ne lotynų kalbos abėcėlę, o lotynų raštą, kuriam priklauso visos lotynų raidės.

Specialieji ženklai gali turėti trumpus, juos vienareikšmiškai įvardijančius pavadinimus, pavyzdžiui,

: COLON,
× MULTIPLICATION SIGN

arba ilgesnius aprašomuosius pavadinimus, pavyzdžiui,

↴ RIGHTWARDS ARROW WITH CORNER DOWNWARDS,
≃ ASYMTOTICALLY EQUAL TO.

Tų pačių ženklų pavadinimai kitomis kalbomis gali turėti kitokio pavidalo pavadinimą, priklausomą nuo jų vartosenos tose kalbose, pavyzdžiui,

~ TILDE (angliškai),
~ Riestinio kirčio ženklas (lietuviškai),
“ LEFT DOUBLE QUOTATION MARK (angliškai, nes anglams tai atidaromosios),
“ Uždaromosios lietuviškos kabutės (lietuviškai).

Lietuvos standartuose (pvz., LST/IEC 8859-13, LST 1590-3) raidžių pavadinimuose priimta vartoti ir didžiąsias, ir mažąsias lietuvių kalbos abėcėlės raides, o taip pat vardus trumpinti neminint to, kas aišku iš konteksto – kad raidė yra lotynų, nes visos standarte minimos raidės priklauso lotynų raštui, pavyzdžiui,

A Didžioji raidė A,
Ą Didžioji raidė Ą,
ą Mažoji raidė ą.

Raidės su diakritiniais ženklais, kurių nėra lietuvių kalbos abėcėlėje, įvardijamos aprašant jų sandarą – išvardijant jų diakritinius ženklus, pavyzdžiui,

Ä Didžioji raidė A su dviem taškais viršuje,
ā Mažoji raidė a su brūkšniu viršuje.

Aprašomaisiais pavadinimais būtų įvardijamos ir savitosios lietuvių kalbos raidės kitomis kalbomis, jeigu jų abėcėlėje tų raidžių nėra, pavyzdžiui, latviai raidę Ė vadintų „E su tašku“ (latviškai), bet raidę Ž vadintų *že,* nes šią raidę ir jie turi.

Raidžių ir kitų ženklų, esančių LST ISO/IEC 8859-13 standarte, pavadinimai pateikti 2-ame priede.

Analogiškai įvardijamos lietuvių kalboje vartojamos kirčiuotos raidės, pavyzdžiui,

À – A su kairiniu kirčiu;
Á – A su dešininiu kirčiu;
Ã – A su riestiniu kirčiu.

Maždaug pusė kirčiuotų raidžių sutampa su kitose kalbose vartojamomis pagrindinių abėcėlių raidėmis. Raidės tos pačios. Skiriasi tik vartosena. Siekiant vienodumo, Lietuvos standartuose visos raidės su ženklais ` ´ ~ , ir tos, kurios lietuvių kalboje nevartojamos, vadinamos analogiškai, kaip kirčiuotos.

Kirčiuotos i ir į raidės (tik mažosios) yra ne tos pačios, kaip kitose kalbose vartojamos tokios pat raidės, tik be taškų. Todėl verčiant į kitas kalbas reikia pateikti šią savybę nusakantį raidžių pavadinimą, pavyzdžiui, í – *mažoji raidė i su tašku ir dešininiu kirčiu*.

Ženklų, taigi ir jų pavadinimų, yra tūkstančiai. Nuolat atsiranda naujų ženklų. Jiems reikia naujų pavadinimų. Gali atsirasti nenumatytų situacijų ir tekti pakoreguoti jau vartojamus pavadinimus.

Nevienareikšmiškumų gali atsirasti ir įvardijant ženklus įvairiomis kalbomis, verčiant iš vienos kalbos į kitą.

Nauja galimybė ženklams identifikuoti atsirado kartu su unikodu. Į unikodą sudėti visų pasaulio kalbų rašto ženklai. Kiekvienas ženklo kodas unikalus. Taigi juo galima vienareikšmiškai identifikuoti ženklą (analogiškai kaip žmogų asmens kodu).

Rašto ženklo kodas nurodomas šešioliktainiu kodo numeriu, o tam, kad parodytume, jog šis užrašas yra unikodo kodas, jo pradžioje rašoma U+, pavyzdžiui,

A U+0041,

Ž U+017D.

Tačiau koduose nėra informacijos apie ženklą, kuri yra sistemingai sudarytuose žodiniuose jų pavadinimuose. Todėl pirminis unikodo šaltinis pateikiamas duomenų bazės pavidalu, kurioje pateikiama informacija apie ženklą, kurios gali prireikti ką nors darant su tuo ženklu.

Informacijos apie ženklo priklausomybę kalbai nėra nei žodiniuose ženklų pavadinimuose, nei unikodo duomenų bazėje, išskyrus ypatingus išimtinus atvejus. Pavyzdžiui, negalime sakyti, kad raidė Ą lietuviška. Ji tiek pat ir lenkiška, gal yra dar kurios nors kalbos abėcėlėje. Kitaip sakant, raidės tautybės neturi.

Raidžių pavadinimuose minima ir pati raidė. Šnekoje (sakytinėje kalboje) ne visas atskirai parašytas raides galima ištarti taip, kad tarimas būtų vienareikšmiškai suprastas, ypač priebalsių ir ilgųjų balsių. Todėl yra dar trumpi šnekai skirti raidžių vardai. Lietuvių kalbos abėcėlės raidžių vardai sudaromi pagal tokią schemą: balsių, išskyrus ilguosius, vardai sutampa su raide, prie ilgųjų pridedamas pažymimasis žodis *ilgoji,* prieš nosinę – *nosinė*, prieš priebalsę arba po jos pridedama balsė *ė* (žr. 2-ą priedą).

3. KODUOTĖS

Nuo pat darbo su tekstais pradžios kompiuteriuose vargstama su rašto ženklų kodavimu. Nuolat jaučiamas kodų stygius ir su juo susiję ženklų aibės ribojimai.

Sakoma, kad visos šios problemos atsiranda dėl kalbų ir raštų įvairovės. Tai tiesa. Bet ta įvairovė buvo žinoma nuo pat kodavimo pradžios, nuo Morzės abėcėlės. Tačiau į tai nebuvo kreipiama dėmesio. Parengiama koduotė vienai kalbai ir tik po to, kai būna pagaminta nemažai įrangos, pradedama galvoti, kaip ją pritaikius kitoms kalboms.

Atsirado unikodas – viena didelė ir universali koduotė, į kurią telpa visų dabar vartojamų kalbų rašto ženklai. Bet problemų vis tiek yra. Dar daug kur, ypač telekomunikacijoje, velkasi keturių dešimtmečių senumo septynbičių koduočių šleifas.

Kai prigaminama daug įrangos su skurdesne koduote, tarkim septynių bitų, į kurią netelpa visi reikalingi ženklai, reikia prie jos prisitaikyti – sugalvoti būdus turtingesnei ženklų aibei išreikšti skurdesnės aibės ženklais ir jų sekomis, rašyti perkodavimo programas ir nuolat jomis naudotis.

Šių problemų galėjo nebūti, jeigu nuo pat teksto apdorojimo pradžios būtų pasirinktas 16 bitų kodavimas. Būtų reikėję dvigubai talpesnės atmintinės – taigi būtų tekę arba mokėti dvigubą kainą, arba pristabdyti sudėtingesnių uždavinių sprendimą, kol kompiuterių atmintinės talpa padvigubės. O ji (tiek operatyviosios, tiek diskinių laikmenų) padvigubėja maždaug kas pusantrų metų. Taigi delsa ne tokia ilga. Bet tais laikais tai atrodė daug. Žmogus linkęs labiau rūpintis šia diena negu rytojumi. Tačiau žiūrint iš šių dienų pozicijų, kaina, kurią teko ir dar tenka mokėti dėl koduočių įvairovės ir atgalinio suderinamumo kur kas aukštesnė.

Praeities nebepakeisi, bet apie tai verta pamąstyti, kad senų klaidų nekartotume dabar.

3.1. Nuo Morzės abėcėlės iki pirmųjų koduočių

Rašto ženklų kodavimas telegrafe buvo naudojamas dar ikikompiuteriniais laikais. Tai Morzės abėcėlė ir telegrafo kodas. XIX a. ketvirtajame dešimtmetyje (1837 m.) telegrafą išrado Samuelis Finlėjus Bryzas Morzė (Samuel Finley Breese Morse). Pradžioje žodžius kodavo skaičiais, po to skaičius paversdavo elektros impulsais ir juos perduodavo laidais. Tai buvo nepatogu, nes reikėjo turėti sunumeruotus žodynus, pagal kuriuos siuntėjas užšifruodavo žodžius skaičiais, o gavėjas iššifruodavo. Vėliau buvo sukurta atskirų rašto ženklų kodavimo sistema, pavadinta Morzės abėcėle.

3.1.1. Morzės abėcėlė

Dėl Morzės abėcėlės autorystės abejojama. Manoma, kad ją iš tikrųjų sukūrė ar bent jau pagrindinį darbą atliko S. Morzės asistentas Alfredas Vailas (Alfred Lewis Vail). Tačiau ji buvo pavadinta visos sistemos, taigi telegrafo kūrėjo vardu.

Raidės, skaitmenys, skyrybos ir kiti ženklai išreiškiami trumpų (žymimų tašku) ir ilgų (žymimų brūkšniu) elektros srovės impulsų seka. Impulsai skiriami trumpais (kaip taškas) tarpais. Raidės žodyje ir skaitmenys skaičiuje skiriami ilgesniais tarpais, žodžiai – dar ilgesniais. Laikoma, kad brūkšnys arba tarpas tarp ženklų tris kartus ilgesnis už tašką, o tarpas tarp žodžių – penkis kartus (kai kuriuose šaltiniuose nurodoma – septynis kartus).

Tai buvo pirmasis abėcėlės ženklų kodavimas diskrečiaisiais signalais, ilgą laiką dominavęs radijo, telegrafo ir kitose ryšio sistemose. Tada kompiuterių dar nebuvo, o vėliau atsiradusiuose kompiuteriuose nebuvo naudojama. Bet šį kodavimą verta panagrinėti iš ženklų kodavimo, taigi ir iš šiuolaikinių kompiuterių, pozicijų.

Ženklų kodai sudaromi iš dviejų elementų – taško ir brūkšnio. Taigi kodavimo požiūriu ji dvejetainė. Tačiau perduodant informaciją svarbų vaidmenį turi tarpai, kuriuos reikėtų laikyti taip pat kodo elementais, nors ir ne taip aiškiai matomais.

Kodai skirtingo ilgio. Kodų skaičius priklauso nuo kodo ilgio. Iš n elementų galima sudaryti 2^n skirtingų kodų. Esant skirtingo ilgio kodams iš visų galimų ilgių galima sudaryti

$$\sum_{i=1}^{n} 2^i$$ kodų (7 lentelė).

Raidžių, skaitmenų ir kai kurių kitų ženklų kodai pateikti 8 lentelėje.

7 lentelė. Kodų skaičius

Kodo elementų skaičius n	Kodų skaičius iš n elementų	Kodų skaičius iš visų (nuo 1 iki n) elementų
1	2	2
2	4	6
3	8	14
4	16	30
5	32	62
6	64	126
7	128	254

Trisdešimties kodų, sudarytų iš 1, 2, 3 ir 4 elementų, pakako 26 anglų kalbos abėcėlės raidėms užkoduoti ir dar keturi kodai liko laisvi. Ekonomijos sumetimais dažniau pasitaikančios raidės buvo užkoduotos trumpesniais kodais.

Skaitmenys koduojami penkių elementų kodais.

Šie 36 kodai sudarė invariantinę Morzės abėcėlės dalį, kuri dažnai vadinama tarptautine Morzės abėcėle.

Invariantinė dalis buvo įtraukiama į įvairių lotyniškus rašmenis vartojančių kalbų Morzės abėcėlės variantus ir papildoma tų kalbų savitosiomis raidėmis, joms priskiriant likusius nepanaudotus kodus.

Raidėms, netilpusioms į keturių elementų kodus (nuo anglų kalbos abėcėlės jų liko tik 4), buvo skiriami ilgesni kodai. Pavyzdžiui, lietuviai raidei Ė paskyrė kodą . . – . . , lenkai kai kurioms jų skyrė net šešių (Ż – – . . – .) ir septynių (Ś . . . – . . .) elementų kodus, nors buvo nepanaudotų trumpesnių kodų. Mat kai kodai įvairaus ilgio, tai pailgintas vienas kitas rečiau vartojamo ženklo kodas problemų nekelia, bet galima išnaudoti tam tikras jų savybes, pavyzdžiui, lengviau juos atpažinti iš labiau išsiskiriančios melodijos juos priimant kaip garsinius signalus.

Tuo pačiu kodu buvo koduojamos panašios įvairių kalbų raidės, pavyzdžiui, kodą . – . – lietuviai ir lenkai paskyrė raidei Ą, vokiečiai – Ä, danai – Æ. Tais laikais kalboms, taigi ir koduotėms atpažinti, priemonių nebuvo, todėl buvo svarbu, kad telegramą nusiuntus į kitą valstybę, kurioje nevartojamos kai kurios siuntėjo kalbos raidės, tos raidės būtų vaizduojamos panašiomis į valstybės, kurioje gyvena gavėjas, kalbos raides. Pavyzdžiui, lietuviško žodžio ĄSA kodas, gautas Vokietijoje pagal vokišką kodą būtų iššifruotas kaip ÄSA. Neteisingai, bet suvokiama, kad čia turi būti kuri nors raidės A modifikacija ir, žvilgterėjus į lietuvių kalbos abėcėlę, galima vienareikšmiškai nustatyti kuri.

8 lentelė. Morzės abėcėlė

Ilgis	Kodas	Ženklas	Kodas	Ženklas
1	·	E	–	T
2	··	I	–·	N
	·–	A	– –	M
3	···	S	–··	D
	··–	U	–·–	K
	·–·	R	– –·	G
	·– –	W	– – –	O
4	····	H	–···	B
	···–	V	–··–	X
	··–·	F	–·–·	C
	··– –	Ū Ü Ŭ	–·– –	Y
	·–··	L	– –··	Z
	·–·–	Ą Ä Æ	– –·–	Q
	·– –·	P	– – –·	Č Ö Ø Ó
	·– – –	J	– – – –	Š Ch Ĥ
5	·····	5	–····	6
	····–	4	–···–	=
	···–·	Ŝ	–··–·	
	···– –	3	–··– –	
	··–··	Ė É Đ Ę	–·–··	Ç Ĉ Ć
	··–·–		–·–·–	
	··– –·	Đ	–·– –·	Ĥ
	··– – –	2	–·– – –	
	·–···		– –···	7
	·–··–	È Ł	– –··–	Ż
	·–·–·	+	– –·–·	Ĝ
	·–·– –		– –·– –	Ñ Ń
	·– –··	Þ	– – –··	8
	·– –·–	À Á Å	– – –·–	
	·– – –·	Ĵ	– – – –·	9
	·– – – –	1	– – – – –	0
6*	·····	.	– –··–·	Ź
	··– –··	?	–····–	–
	·–··–·	“	–·–·–·	;
	·–·–·–	,	– –··– –	!
	·– – – – ·	‘	– – – ···	:
7**	···–···	Ś	···– –··	ß

* Iš viso 6 elementais galima užkoduoti 64 ženklus (lentelėje reikėtų 32 eilučių). Lentelėje parodyta tik dalis jų, panaudotų vienai raidei ir svarbesniems skyrybos ženklams koduoti.

** Iš viso 7 elementais galima užkoduoti 128 ženklus. Lentelėje parodyti tik du kodai, panaudoti raidėms koduoti.

Dauguma skyrybos ir kitų ženklų koduojami 6 elementais, nors dar yra laisvų 5 elementų kodų. Jų kodavimas skirtingose kalbose įvairuoja, lentelėje pateikti tik dalis lietuviškoje Morzės abėcėlėje esančių skyrybos ženklų.

Kitokius rašmenis (kirilicą, graikų) vartojančios kalbos turi savas Morzės abėcėles. Jų raidės dažniausiai koduojamos tais pačiais kodais kaip tas raides atitinkančios lotynų raidės pagal tarimą. Pavyzdžiui, rusiškai raidei A paskirtas toks pats kodas kaip ir lotynų A, rusiškai Б – kaip lotynų B ir t. t. Todėl jeigu rusiškas koduotas tekstas pateks vien lotynų abėcėlę vartojančiam gavėjui, tai jis bus suprantamas kaip transliteruotas į lotynų rašmenis. Tai dera ir su lietuvių kalbos abėcėle (Ą – Я; Č – Ч; Ė – Э; Ū – Ю).

Japonų katakanos Morzės abėcėlėje tik skaitmenų kodai sutampa su europietiškais. Visi likusieji 1–5 elementų kodai panaudoti katakanos ženklams.

Kalboms, vartojančios ideografinius rašmenis, dėl didelio ženklų skaičiaus buvo kebliau. Jos hieroglifus sunumeruodavo ir persiųsdavo jų numerius. Kinai hieroglifus koduodavo keturženkliais skaičiais. Pavyzdžiui, frazė 中文信息 (Čung-ven sin-si), reiškianti „Informacija Kinijoje“ būtų pavaizduota skaičiais 0022 2429 0207 1873. Skaitmenų daug, jų kodai ilgi. Todėl buvo parengtas ekonomiškesnis hieroglifų kodavimas raidėmis. Vienas hieroglifas koduojamas trijų raidžių grupe: AAA, AAB ir t. t.

Dabar ryšiuose Morzės abėcėlė beveik nevartojama, tačiau neliko pamiršta. Tarptautinė telekomunikacijų sąjunga ITU (angl. *International Telecommunication Union*) 2004 metais ją papildė ženklo @ kodu: . – – . – . , kad būtų galima perduoti elektroninio pašto adresą.

Mobiliųjų telefonų programinės įrangos lokalizuotojams su Morzės abėcėle gali tekti susidurti ir tiesiogiai, nes pastaruoju metu telefonų gamintojai ja susidomėjo kaip labai paprastu ir pakankamai sparčiu trumpųjų žinučių įvedimo būdu, tinkamu neįgaliesiems (tokią funkciją turi keletas „Nokia“ mobiliųjų telefonų modelių). Žinutės pavertimas Morzės abėcėlės ženklais, pateikiamais balsu arba vibracija tinka laisvų rankų įrangai.

3.1.2. Telegrafo kodas

1870 metais prancūzų inžinierius ir išradėjas Žanas Morisas Emilis Bodo (Jean-Maurice-Émile Baudot) sukūrė ir 1874 metais užpatentavo penkių elementų (bitų) kodą. Vėliau kodas buvo ne kartą tobulinamas. Paskutinis ir daugiausiai naudotas variantas yra žinomas kaip Tarptautinis telegrafo kodas Nr. 2, trumpiau – ITA2.

Fiksuoto ilgio telegrafo kodas automatizuotam informacijos perdavimui geriau tiko negu Morzės abėcėlė. Tekstą buvo galima iš anksto paruošti jį pramušant į perfojuostą (popierinę juostelę, padalintą į penkias eilutes – po vieną kiekvienam ženklo bitui).

Siekiant ekonomiškumo buvo pasirinktas kuo trumpesnis, penkių dvejetainių ženklų, kodas. Iš penkių dvejetainių elementų galima sudaryti $2^5 = 32$ kodus. Vienos kalbos abėcėlei pakaktų. Bet dar yra skaitmenys ir keletas kitų būtiniausių ženklų. Todėl buvo padaryti du registrai: vienas raidžių, kitas skaitmenų. Į skaitmenų registrą sudėti ir kiti ženklai. Registrai perjungiami valdymo kodais, kurie lentelėje pažymėti LTRS (angl. *letters* – raidės) ir FIGS (angl. *figures* – skaitmenys).

Perjungdami registrus gauname lyg ir dvi kodų lenteles (3 ir 4 pav.).

Abiejose lentelėse ženklai sunumeruoti aštuonetainiais skaičiais nuo 00 iki 37. Stulpeliai numeruojami pirmuoju kodo numerio skaitmeniu (po jų prirašyti nuliai, kad į jų vietą skaitydami kodus galėtume įterpti eilučių numerius), eilutės – antruoju (todėl prieš juos prirašyti nuliai). Ženklo kodas gaunamas sudėjus jo stulpelio ir eilutės numerius, pavyzdžiui, raidės N kodas yra 10 + 04 = 14. Eilučių numerius rašome dešinėje pusėje, nes taip paprasčiau rasti kodus: pirmiau skaitome stulpelio numerį viršuje, po to eilutės numerį dešinėje. Taip numeruosime ir kitų kodų lentelių langelius. Tiktai, kai kodai bus ilgesni (7 arba 8 bitų), vartosime šešioliktainius skaitmenis.

Panaudoję du registrus gauname dvigubai daugiau kodų. Bet ženklų skaičius ne visai padvigubėja – dalį kodų reikia skirti registrams perjunginėti. Teoriškai kiekvienam registrui pakaktų vieno kodo: raidžių registre skaitmenų registro kodo (perjungimas iš raidžių registro vėl į raidžių registrą neturi prasmės), skaitmenų registre –

00	10	20	30	
NULL 0	CR 8	T 16	O 24	00
E 1	D 9	Z 17	B 25	01
LF 2	R 10	L 18	G 26	02
A 3	J 11	W 19	FIGS 27	03
SP 4	N 12	H 20	M 28	04
S 5	F 13	Y 21	X 29	05
I 6	C 14	P 22	V 30	06
U 7	K 15	Q 23	LTRS 31	07

3 pav. ITA2 kodo raidžių registras

00	10	20	30	
NULL 0	CR 8	5 16	9 24	00
3 1	ENQ 9	+ 17	? 25	01
LF 2	4 10	) 18	& 26	02
– 3	BELL 11	2 19	FIGS 27	03
SP 4	, 12	# 20	. 28	04
' 5	! 13	6 21	/ 29	05
8 6	: 14	0 22	; 30	06
7 7	(15	1 23	LTRS 31	07

4 pav. ITA2 kodo skaitmenų registras

raidžių registro kodo. Tačiau tai būtų nepatikima – padarius registrų perjungimo klaidą visas tolesnis tekstas būtų koduojamas neteisingai. Todėl jų kodai padubliuoti abiejuose registruose.

Abiejuose registruose dubliuojami ir kiti svarbesni valdymo kodai, kurių klaidingas panaudojimas galėtų sudarkyti tekstą:

NULL – tuščia vieta;

LF (angl. *line feed*) – eilutės patraukimas;

SP (angl. *space*) – tarpas;

CR (angl. *carriage return*) – grįžimas į eilutės pradžią.

Jų simboliai pavaizduoti juodame fone.

Taigi antrasis registras ženklų skaičių ne padvigubino, o papildė tik 24 ženklais ($2 \times 2 = 4$ kodai panaudoti registrams perjungti ir dar 4 kodai, skirti kitiems valdymo simboliams, dubliuojami abiejuose registruose).

Trys skaitmenų registro kodai (jų simboliai pavaizduoti pilkame fone) skirti numatytiesiems ITA2 kodo ženklams ! # &. Konkrečių kalbų koduotėse vietoj jų gali būti kiti ženklai. Štai keletas pavyzdžių:

anglų – % £ @,

švedų – Å Ö Ä,

vokiečių – Ä Ü Ö.

Kodas faktiškai tiko tik anglų kalbai ir keletui kitų kalbų, kurių abėcėlėse buvo nedaugiau kaip trys raidės, kurių nėra anglų kalbos abėcėlėje. Todėl jį vadinti tarptautiniu nelabai tinka.

3.1.3. Ženklų kelias į kompiuterį

Tekstams koduoti buvo galima pasinaudoti telegrafo kodu, o kartu su juo ir telegrafe naudojama aparatūra perfojuostoms pramušti ir skaityti. Tačiau telegrafo kodas turėjo ir trūkumų.

Veiksmus su tekstais kompiuteryje patogiau atlikti, kai ženklų kodai išdėstyti pagal jais užkoduotų ženklų natūralią eilę (raidžių – pagal abėcėlę, skaitmenų – didėjančiai). Todėl telegrafo kodas buvo naudojamas tik duomemims įvesti iš perfojuostos arba jiems išvesti ir tokiu būdu panaudoti perfojuostą kaip išorinę duomenų laikmeną. Vidiniam duomenų kodavimui buvo kuriamos kompiuteriams tinkamesnės koduotės. Kaip pavyzdį pateiksime vieną tokią koduotę, neblogai atspindinčią tų laikų koduočių įvairovę (5 ir 6 pav.).

00	10	20	30	
SP 0	H 8	O 16	U 24	00
A 1	I 9	P 17	V 25	01
B 2	. 10	Q 18	W 26	02
C 3	J 11	R 19	X 27	03
D 4	K 12	FS 20	Y 28	04
E 5	L 13	LS 21	Z 29	05
F 6	M 14	S 22	NL 30	06
G 7	N 15	T 23	DEL 31	07

5 pav. Kompiuterio „English Electric Deuce“ koduotės raidžių registras

00	10	20	30	
0 0	8 8	16 16	 24	00
1 1	9 9	17 17	+ 25	01
2 2	10 10	18 18	– 26	02
3 3	11 11	19 19	. 27	03
4 4	12 12	FS 20	EON 28	04
5 5	13 13	LS 21	 29	05
6 6	14 14	 22	NL 30	06
7 7	15 15	 23	DEL 31	07

6 pav. Kompiuterio „English Electric Deuce“ koduotės skaitmenų registras

Skaitmenys 0–9 išdėstyti lentelės pradžioje iš eilės. Jų kodai (aštuonetainiai 00–11) lygūs patiems skaitmenims. Todėl kompiuteris gali atlikti aritmetinius veiksmus tiesiogiai su skaitmenų kodais. Toks skaitmenų kodavimas būdingas ir daugumai kitų penkių bitų koduočių.

Šioje koduotėje neišvengta ir egzotikos. Kodus turėjo ir dviženkliai skaičiai 10–19. Tai Didžiosios Britanijos pinigų sistemos atspindys. Iki 1971 metų svaras sterlingų turėjo 20 šilingų, o šilingas – 12 pensų. Taigi buvo atskiras kodas kiekvienam šilingui (ir kiekvienam pensui). Kitose angliškų kompiuterių koduotėse (LEO, EMI 2400) buvo „pagerbti“ tik pensai – atskirus kodus turėjo tik skaičiai 10 ir 11.

3.1.4. Šešių bitų koduotės

Rimtesniam darbui su tekstu penkių bitų kodai nelabai tiko dėl kodų skaičiaus ribotumo. Padėties negelbėjo ir ženklų aibės praplėtimas antru registru.

Kompiuterijai ypač nepatogi buvo registrų sistema. Esant laisvai prieigai prie duomenų nėra informacijos, kuriam registrui priklauso atskirai paimtas kodas, taigi ir kuris ženklas juo užkoduotas. Todėl telegrafo kodas ir penkių bitų kodavimas apskritai kompiuteriuose buvo tik epizodinis reiškinys. Buvo kuriami nauji, kompiuteriams tinkamesni kodai.

Pirmasis žingsnis – link 6 bitų koduočių. Šešiais bitais galima užkoduoti $2^6 = 64$ ženklus. Į tokią koduotę gali tilpti raidės, skaitmenys, skyrybos ženklai. Buvo galima nebenaudoti registrų.

Šešių bitų koduotės buvo naudojamas didžiuosiuose kompiuteriuose (tuomet lietuviškai vadintuose elektroninėmis skaičiavimo mašinomis). Spausdinimo įrenginiai tuomet buvo gana primityvūs ir spausdindavo tik vieno registro raides, dažniausiai didžiąsias. Todėl ir kompiuteriniuose dokumentuose buvo vartojamos vien didžiosios raidės.

Tais laikais kompiuteriai buvo labai įvairūs. Šešių bitų koduotes kūrė kompiuterių gamintojai, dažniausiai vis kitokią kiekvienai kompiuterių serijai (7 ir 8 pav.).

Šešių bitų koduotėse buvo nemažai matematikai ir programavimo kalboms reikalingų ženklų: visų keturių operacijų ženklai ($+ - \times /$), loginių operacijų ($\neg \wedge \vee \equiv$), visų palyginimo operacijų ($< \leqslant = \neq > \geqslant$) ženklai. Tai rodo, kad pirmosios abėcėlių koduotės buvo labiausiai reikalingos programuotojams ir matematikams.

Raidės buvo tik iš anglų kalbos abėcėlės, be galimybės įtraukti kitų kalbų raides. Už borto liko ir mažosios raidės. Tai rodo, kad tuomet natūralių kalbų tekstai kompiuteriuose dar buvo retenybė.

Randantis poreikiui apdoroti įvairių kalbų tekstus, keitėsi ir kompiuteriai. Buvo orientuojamasi į aštuonių bitų koduotes, o kompiuterio žodžio ilgis buvo parenkamas dalus iš 8. Vėliau pereita prie adresuojamų aštuonbičių baitų.

00	10	20	30	40	50	60	70	
@ 0	C 8	K 16	S 24	) 32	* 40	0 48	8 56	00
[1	D 9	L 17	T 25	– 33	(41	1 49	9 57	01
] 2	E 10	M 18	U 26	+ 34	" 42	2 50	' 58	02
# 3	F 11	N 19	V 27	< 35	: 43	3 51	; 59	03
Δ 4	G 12	O 20	W 28	= 36	? 44	4 52	/ 60	04
SP 5	H 13	P 21	X 29	> 37	! 45	5 53	. 61	05
A 6	I 14	Q 22	Y 30	& 38	, 46	6 54	¤ 62	06
B 7	J 15	R 23	Z 31	$ 39	\ 47	7 55	≠ 63	07

7 pav. Kompiuterio „Univac 1100" koduotė

00	10	20	30	40	50	60	70	
: 0	H 8	P 16	X 24	5 32	/ 40	≡ 48	↑ 56	00
A 1	I 9	Q 17	Y 25	6 33	(41	[49	} 57	01
B 2	J 10	R 18	Z 26	7 34	) 42	] 50	< 58	02
C 3	K 11	S 19	0 27	8 35	$ 43	% 51	> 59	03
D 4	L 12	T 20	1 28	9 36	= 44	≠ 52	≤ 60	04
E 5	M 13	U 21	2 29	+ 37	SP 45	{ 53	≥ 61	05
F 6	N 14	V 22	3 30	– 38	, 46	∨ 54	⌐ 62	06
G 7	O 15	W 23	4 31	* 39	. 47	∧ 55	; 63	07

8 pav. Kompiuterio „CDC 6000" koduotė

Aštuoni bitai buvo patogūs ir tuo, kad buvo galima turėti atskirą ekonomišką kodavimą skaičiams, kai duomenys vien skaitiniai: į baitą dėti po du skaitmenis. Iš pradžių šiuo kodavimu buvo žavimasi, bet pingant kompiuteriams toks kodavimas pasidarė nebeaktualus.

Į lotyniškiems rašmenims skirtas šešių bitų koduotes buvo įtrauktos tik anglų kalbos abėcėlės raidės nepaliekant vietos kitų kalbų, vartojančių lotyniškus rašmenis, raidėms. Tai buvo žingsnis atgal lyginant su Morzės abėcėle ir telegrafo kodu. Jį galima paaiškinti tuo, kad tų laikų kompiuteriai tekstams apdoroti buvo beveik nenaudojami. Pirmieji tekstus pradėjo naudoti matematikai ir programuotojai, kuriems buvo reikalingesni matematikos ženklai.

Dėl paminėtų trūkumų šešių bitų kodavimas egzistavo neilgai. Projektuojant kompiuterius buvo pereinama prie aštuonių bitų, bet telekomunikacijose ir kitur dar ilgam išliko septynių bitų kodavimas. Priežastys dvejopos: 1) siekiant patikimumo su kiekvienu ženklu buvo siunčiamas kontrolinis jo kodo lyginumo bitas, todėl ženklo kodui iš aštuonių bitų likdavo septyni; 2) anglų kalbos tekstams pakako septynių bitų koduotės.

3.2. Septynių bitų koduotės

Septyniais bitais galima užkoduoti 128 ženklus. Tai gana daug, į koduotę galima sudėti skaitmenis, didžiąsias ir mažąsias raides, nemažai specialiųjų ženklų.

Labiausiai žinoma ir turėjusi didelę įtaką tolesnėms koduotėms (8 bitų ir unikodui) yra giminingų koduočių pora: ASCII ir ISO 646.

3.2.1. ASCII koduotė

1963 metais JAV standartų asociacija ASA (angl. *American Standards Association*) standartizavo septynių bitų koduotę ASCII-1963. Joje buvo didžiosios anglų kalbos abėcėlės raidės, skaitmenys, nemažai specialiųjų ženklų, valdymui skirtų kodų, bet buvo numatyta vieta ir mažosioms raidėms. Šią koduotę galima laikyti ASCII koduotės prototipu arba tiesiog pirmuoju jos leidimu, kadangi visi didžiųjų raidžių ir skaitmenų kodai, o taip pat beveik visi joje buvusių specialiųjų ženklų kodai išliko tie patys kituose jos leidimuose, vėliau buvo perkelti į 8 bitų koduotes ir net unikodą.

Antrajame standarto leidime (1967 m.) jau buvo ir mažosios raidės. Vėliausias, dabar naudojamas, leidimas paskelbtas 1986 metais (ASCII, 1986) (9 pav.). Formaliai – tai ANSI X3.4 standartas. Tačiau koduotei prigijo ASCII pavadinimas. Langelio apačioje mažais skaičiais užrašytas dešimtainis ženklo kodas. Pavyzdžiui, raidės N dešimtainis kodas yra 78 (jos šešioliktainis kodas 4E). Dešimtainiai kodai naudojami renkant klaviatūroje nesančius ženklus (žr. 3 priedą). Toliau įvardydami ženklus naudosime šešioliktainius kodus, o kartais skliaustuose nurodysime ir dešimtainius.

ASCII koduotėje iš viso 128 kodai:

- 94 spausdinamieji rašmenys;
- 1 tarpo ženklas (kodas 20 (32), koduotėse žymimas raidėmis SP);

00	10	20	30	40	50	60	70	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	0F

9 pav. ASCII koduotė

- 1 niekinis ženklas (kodas 7F (127), koduotėse paliekamas tuščias langelis arba žymimas DEL);
- 32 valdymo ženklai (kodai 00–1F).

Spausdinamieji ženklai – tai didžiosios ir mažosios anglų kalbos abėcėlės raidės, skaitmenys ir kiti rašto ženklai, turintys piešinius. Todėl jie dar vadinami grafiniais ženklais.

Visi kiti ženklai piešinių neturi. Koduotėse jie įvardijami simboliniais pavadinimais – angliškomis santrumpomis (pvz., SP), tekste – įprastais pavadinimais (pvz., *tarpo ženklas*).

Tarpo ženklas – tai tuščia vieta eilutėje žodžiams skirti. Jį galima laikyti rašto ženklu.

Niekinis ženklas yra istoriškai išlikęs naikinimo ženklas, vartotas pramušant perfojuostas. Jeigu pramušant suklystama, tai net ir iškart pastebėtą klaidą ištaisyti sudėtinga – nėra kaip užklijuoti jau pramuštas skylutes. Todėl užuot klaidą taisius, neteisingas kodas panaikinamas pramušant skylutes visose likusiose kodo vietose. Gaunamas kodas 7F, sudarytas vien iš dvejetainių vienetų. Skaitant perfojuostą toks kodas praleidžiamas lyg ten nieko nebūtų buvę. Dabar šis kodas nebenaudojamas, todėl ir jo langelis koduotėse dažniausiai paliekamas tuščias (anksčiau buvo žymimas simboliu DEL).

Penkių takelių perfojuostoje visus vienetus turėjo raidžių registro kodas LTRS. Bet jis buvo netiesiogiai vartojamas ir taisymui. Jeigu klaida padaryta pramušant raidės kodą, tai pakartotinas raidžių registro kodas nieko nereiškia, o jei skaitmenų registro ženklus, pataisius reikia grįžti į skaitmenų registrą. Niekinis ženklas išliko ir tolesnėse koduotėse, netgi unikode (U+007F).

Valdymo ženklais perduodamos komandos tekstą apdorojančiam įrenginiui.

Tekstai sudaromi iš spausdinamųjų ženklų ir tarpų. Be to, juose gali būti vartojami ir šie valdymo ženklai:

- horizontalusis tabuliavimas (09). Prilygsta keliems tarpams, priklausomai nuo jo vietos teksto eilutėje, ir vartojamas grynojo teksto lentelių, sudarytų vien iš teksto ženklų, stulpeliams lygiuoti;
- vertikalusis tabuliavimas (1B). Prilygsta vienam ar daugiau intervalų tarp eilučių ir skirtas vertikaliam teksto eilučių lygiavimui. Vartojamas retai;
- eilutės patraukimo ženklas (0A) ir grįžimas į eilutės pradžią (0D). Vartojami grynajam tekstui skaidyti į eilutes. Ten, kur reikia tekstą kelti į naują eilutę, įterpiami abu ženklai arba kuris nors vienas priklausomai nuo operacinės sistemos:

- 0A – „Unix“ ir „Linux“;
- 0D – „Mac OS“;
- 0A ir 0D ženklų pora – DOS ir „Windows“.

ASCII koduotėje yra ne tik didžiosios, bet ir mažosios anglų kalbos abėcėlės raidės. Todėl jos pakanka anglų kalbai ir keletui kitų kalbų, vartojančių anglų kalbos abėcėlę arba jos poaibį (lotynų, kai kur minima ir havajiečių, bet iš tikrųjų šios kalbos abėcėlėje yra raidžių su diakritiniais ženklais). Ši koduotė JAV populiari iki šiol. Netgi grynasis tekstas angliškoje literatūroje dažnai neteisingai įvardijamas santrumpa ASCII.

Kitoms kalboms, turinčioms kitokių raidžių, ši koduotė netiko. Tačiau kompiuterių pramonės lyderis buvo JAV, čia buvo gaminama daug 7 bitams pritaikytos įrangos, ypač skirtos telekomunikacijoms, kuri plito ir į kitas šalis. Todėl kitoms valstybėms reikėjo savų 7 bitų koduočių.

3.2.2. ISO 646 koduotė

Siekiant harmonizuoti įvairių kalbų koduotes, 1965 metais buvo išleistas ECMA-6 standartas, o 1972 metais ISO/IEC 646 tarptautinio standarto pirmasis leidimas. Abu standartai apibrėžia tą pačią koduotę. Nuo ASCII koduotės skiriasi tuo, kad ISO standarte deklaruojama, jog 12 kodų, kuriais koduojami mažiau reikalingi specialieji ženklai, gali būti panaudoti savitosioms įvairių kalbų raidėms arba kitiems tose kalbose svarbiems ženklams. Keičiamųjų kodų ženklai 10 paveiksle pavaizduoti pilkame fone.

Šios koduotės pagrindu kiekviena valstybė galėjo sukurti savą ISO 646 koduotės variantą, jį užregistruoti ISO organizacijoje ir gautu IR numeriu identifikuoti savo koduotę. Daugelis valstybių koduotę įteisino ir savu nacionaliniu standartu. Keleto valstybių nacionalinių koduočių variantuose panaudoti keičiamieji ženklai pateikti 9 lentelėje.

Dviejų ženklų keitimo galimybės ribotos – galima pasirinkti tik vieną variantą iš dviejų:

23 (35) pozicijoje # arba £;

24 (36) pozicijoje $ arba ¤.

INV – invariantinė koduotė, kurioje yra ženklai, bendri visoms iš jos gaunamoms nacionalinėms koduotėms (su 12 tuščių vietų, paliktų nacionalinių koduočių reikmėms).

IRV – standarto variantas, kuriame nacionalinių koduočių reikmėms skirtos vietos užpildytos numatytaisiais ženklais.

Prancūzijos 7E (126) ženklas yra du taškai viršuje (U+00A8).

00	10	20	30	40	50	60	70	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	0F

10 pav. ISO 646 koduotė

Nyderlandų 7C (124) ženklas yra raidė f su uodega (U+0192).

Vengrijos 7E (126) ženklas yra dvigubas akūtas (U+02DD).

Savą standarto variantą turi netgi Jungtinė Karalystė – ASCII koduotėje nebuvo svaro sterlingų ženklo. Vien dėl savo valiutos ženklų savus lotyniškų rašmenų koduotės variantus turėjo Japonija, Kinija ir Korėja (į lentelę jos neįtrauktos).

Kuriant ISO 646 koduotę buvo ginčų dėl numatytojo valiutos ženklo. ASCII koduotėje buvo dolerio ženklas $. Tai suprantama, nes ši koduotė JAV nacionalinė. Bet standartas ISO/IEC 646 tarptautinis. Todėl IRV koduotėje iš pradžių (nuo 1972 m.) buvo sutarta numatytuoju laikyti neutralų valiutos ženklą ¤ ir vadinti jį tiesiog valiutos ženklu. Bet nuo 1991 metų (ISO 646 standarto trečiojo leidimo) jis buvo pakeistas dolerio ženklu.

9 lentelė. Nacionaliniai ISO 646 standarto variantai

Šešioliktainis kodas	**23**	**24**	**40**	**5B**	**5C**	**5D**	**5E**	**60**	**7B**	**7C**	**7D**	**7E**	**Nacionalinis standartas**	**ISO IR**
Dešimtainis kodas	**35**	**36**	**64**	**91**	**92**	**93**	**94**	**96**	**123**	**124**	**125**	**126**		
ISO646-INV														170
ISO 646-IRV	#	$	@	[	\	]	^	`	{	|	}	~		2
JAV (ASCII)	#	$	@	[	\	]	^	`	{	|	}	~	ANSI X3.4-1986	6
Airija	£	$	Ó	É	Í	Ú	Á	ó	É	í	ú	á	IS 433:1996	207
Danija	#	$	@	Æ	Ø	Å	^	`	Æ	ø	å	~	DS 2089	
Italija	£	$	§	°	ç	é	^	ù	À	ò	ù	ì		15
Islandija	#	¤	Ð	Þ	\	Æ	Ö	ð	þ	|	æ	ö		
Ispanija	#	$	·	¡	Ñ	Ç	¿	`	´	ñ	ç	¨		85
Ispanija*	#	$	§	¡	Ñ	¿	^	`	°	ñ	ç	~		17
Buv. Jugoslavija	#	$	Ž	Š	Đ	Ć	Č	ž	Š	đ	ć	č	JUS I.B1.102	141
Jungtinė Karalystė	£	$	@	[	\	]	^	`	{	|	}	~	BS 4730	4
Kanada	#	$	à	Â	ç	ê	î	ô	é	ù	è	û	CSA Z243.4	121
Kanada**	#	$	à	Â	ç	ê	É	ô	é	ù	è	û	CSA Z243.4	122
Kuba	#	$	¤	¡	Ñ	]	¿	`	°’	ñ	[	¨	NC_NC00-10:81	151
Lenkija	#	zł	ę	Ź	\	ń	ś	ą	Ó	ł	ż	ć	BN-74/3101-01	
Malta	#	$	@	Ġ	ż	ħ	^	ċ	Ġ	Ż	Ħ	Ċ		
Nyderlandai	£	$	¾	Ÿ	½	|	^	`	¨	*f*	¼	´		
Norvegija	#	$	@	Æ	Ø	Å	^	`	Æ	ø	å	~	NS 4551-1	60
Prancūzija	£	$	à	°	ç	§	^	`	é	ù	è	¨	NF Z 62-010	69
Portugalija	£	$	´	Ã	Ç	Õ	^	`	Ã	ç	õ	~		84
Portugalija*	#	$	§	Ã	Ç	Õ	^	`	Ã	ç	õ	°		16
Suomija	#	$	@	Ä	Ö	Å	Ü	é	Ä	ö	å	ü		10
Švedija	#	$	@	Ä	Ö	Å	^	`	Ä	ö	å	~	SEN 850200 B	10
Švedija***	#	$	É	Ä	Ö	Å	Ü	é	Ä	ö	å	ü	SEN 850200 C	11
Šveicarija	ù	$	à	É	ç	ê	î	ô	Ä	ö	ü	û		
Vengrija	#	¤	Á	É	Ö	Ü	^	á	É	ö	ü	˝	MSZ 7795.3	86
Vokietija	#	$	§	Ä	Ö	Ü	^	`	Ä	ö	ü	ß	DIN 66003	21
T.61****	#	¤	@	[		]				|				102

* Tarptautinis.
** Po rašybos reformos.
*** Užsienietiškiems vardams užrašyti.
**** ITU/CCITT rekomendacija tekstams dokumentų, perduodamų telefono linijomis.

Buvo pakeistas dar vienas ženklas – pertraukta stati linija ¦ stačiu brūkšniu |. Tarptautinė koduotė ISO 646 IRV tapo tapati JAV nacionalinei koduotei ASCII.

Daugelis specialiųjų ženklų, įtrauktų į ASCII koduotę ir iš jos patekę į ISO 646 koduotę, nėra patys reikalingiausi. Apie jų atsiradimą bei keitimą daug informacijos pateikia Jukka Korpela (Korpela, 2006).

Kai kurios valstybės, kurių kalbos turi nedaug savitųjų raidžių, į nacionalines koduotes įtraukė ir vieną kitą kitos kalbos raidę arba ženklų, kurių piešiniai primena diakritinius ženklus, kad su jais būtų galima išgauti raides su diakritiniais ženklais.

Iš 12 keičiamų kodų tik 10 galima panaudoti papildomoms raidėms. Taigi yra galimybė koduotę papildyti tik 5 raidėmis (5 didžiosiomis ir 5 mažosiomis). Tai mažai ir gali tenkinti tik kalbas, turinčias nedaug raidžių, besiskiriančių nuo anglų kalbos abėcėlės (pvz., vokiečių). Kalboms, turinčioms daugiau raidžių, teko arba standartizuoti tik svarbesnes raides, arba atsisakyti didžiųjų raidžių (ką padarė lenkai).

Lietuva šio standarto varianto neturėjo. Viena, normaliam darbui lietuvių kalbai reikia 18 papildomų kodų pozicijų, o yra tik 10. Įtraukus tik dalį raidžių su diakritiniais ženklais standartas vis tiek nebūtų visavertis. Nebent buvo galima sekti lenkų pavyzdžiu – į koduotę įtraukti tik mažąsias raides. Antra, tuo metu, kai buvo kuriami ISO 646 koduotės nacionaliniai variantai Lietuva dar priklausė Sovietų Sąjungai, kur buvo reikalaujama vartoti rusiškas koduotes.

Septynių bitų koduote faktiškai galima užkoduoti tik vienos kalbos abėcėlės raides ir tokios, kuri turi ne daugiau kaip penkias savitąsias raides. Todėl beveik tuo pat metu buvo rengiamos ir aštuonių bitų koduotės. Ir pats skaičius 7 nelabai tinkamas. Septynių bitų kodas neracionaliai užrašomas tiek aštuonetaine sistema (reikia trijų skaitmenų, o pirmasis skaitmuo gali būti tik 0 arba 1), tiek šešioliktaine sistema (reikia dviejų skaitmenų, o pirmais gali būti tik iš intervalo 0...7). Tačiau septynių bitų koduotės išliko ilgą laiką. Mat telekomunikacijose, kurios tuo metu buvo mažai patikimos, kiekvieno baito vienas bitas buvo skiriamas perduodamo baito lyginumo kontrolei. Taigi iš tikrųjų po kompiuterį ir ryšio kanalus cirkuliavo baitai, t. y. aštuonių bitų kodai, bet iš jų tik septyni buvo panaudojami duomenims (tekstui) koduoti.

ISO 646 INV koduotė turėjo įtaką ir kitiems ISO standartams. Juose (pvz., klaviatūros standarte ISO /IEC 9995) privalomais buvo laikomi tik ženklai, esantys invariantinėje dalyje.

ASCII koduotė turėjo įtaką kompiuterijos pramonei, nes JAV šioje srityje dominavo. Todėl ASCII koduotė buvo daugiau žinoma pasauliui. Jos pagrindu sudarytos aštuonių bitų koduotės, ji tiesiogiai įtraukta ir į unikodą.

Prieštaravimą tarp šių koduočių galima pailiustruoti ženklu @. ISO 646 INV jo nėra. Taigi jis neprivalomas ir kituose standartuose, kurie sudaryti septynbitės koduotės pagrindu. Tačiau vėliau, kai buvo priimtas ISO 646 standartas, ir žinant, kad šio ženklo nėra ISO koduotės invariantinėje dalyje, šis ženklas pradėtas naudoti kiekviename elektroninio pašto adrese.

3.3. Aštuonių bitų koduotės

Svarbiausias aštuonių bitų koduotes galima suskirstyti į tokias grupes:

1. Didžiųjų kompiuterių.
2. Asmeninių kompiuterių:
   - DOS sistemos,
   - ISO 8859 ir „Linux“ sistemos,
   - „Windows“ sistemos,
   - „Mac OS“ sistemos.

Visų keturių asmeninių kompiuterių grupių koduočių pagrindas yra 7 bitų ASCII koduotė. Pridėjus prie šios koduotės vieną (aštuntą) bitą gaunama 8 bitų koduotė. Viskas, kas yra pirmoje (pagrindinėje) pusėje, išlieka iš ASCII koduotės, o antroji (papildomoji) pusė skiriama naujiems ženklams.

Didžiųjų kompiuterių koduotės kildinamos taip pat iš vieno prototipo – EBCDIC koduotės, kurią sukūrė bendrovė IBM 1962–1963 metais ir įdėjo į tuo metu populiarius didžiuosius kompiuterius „IBM System/360“. (Šie kompiuteriai buvo „kopijuojami“ ir gaminami Sovietų Sąjungoje bei kitose komunistinio bloko šalyse. Tai populiari „Vieningosios sistemos“ (rus. „Единая система“) kompiuterių serija, sutrumpintai žymima VS (rus. EC)).

Iš šios koduotės buvo galima gauti įvairioms kalboms tinkamas koduotes palikus nepakeistą invariantinę kodų dalį su skaitmenimis, anglų kalbos abėcėlės raidėmis ir svarbesniais specialiaisiais ženklais, o į paliktas laisvas vietas įdėjus savituosius kalbos rašmenis.

Latvijos standarte RST 0040-90 buvo apibrėžtas EBCDIC variantas, turintis visas latvių ir lietuvių kalbų abėcėlių raides (Grigas, 1998).

3.3.1. Koduotės įvairiose operacinėse sistemose

Koduotės DOS operacinėje sistemoje

Nuo 1981 metų IBM kompanija pradėjo gaminti IBM PC kompiuterius su 8 bitų koduote IBM 437 (11 pav.), dar vadinama PC 437. Tai buvo ASCII koduotė, papildyta stambesnių Europos valstybių kalbų – prancūzų, ispanų, vokiečių ir kt. – raidėmis, specialiaisiais, daugiausia matematikos bei pseudografikos ženklais.

Siekiant dermės su 7 bitų ASCII koduote (žr. 9 pav.), į 8 bitų koduotę įtraukiama visa ASCII koduotė, t. y. pirmoji 8 bitų kodų lentelės pusė (0–127 kodai) sutampa su ASCII kodais, todėl IBM 437 koduotė kartais dar vadinama išplėstąja ASCII koduote.

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	Ç 128	É 144	á 160	▒ 176	└ 192	╨ 208	α 224	≡ 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	ü 129	æ 145	í 161	▒ 177	┴ 193	╤ 209	ß 225	± 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	é 130	Æ 146	ó 162	▓ 178	┬ 194	╥ 210	Γ 226	≤ 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	â 131	ô 147	ú 163	│ 179	├ 195	╙ 211	π 227	≤ 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	ä 132	ö 148	ñ 164	┤ 180	─ 196	╘ 212	Σ 228	⌠ 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	à 133	ò 149	Ñ 165	╡ 181	┼ 197	╒ 213	σ 229	⌡ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	å 134	û 150	ª 166	╢ 182	╞ 198	╓ 214	µ 230	÷ 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	ç 135	ù 151	º 167	╖ 183	╟ 199	╫ 215	τ 231	≈ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	ê 136	ÿ 152	¿ 168	╕ 184	╚ 200	╪ 216	Φ 232	° 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	ë 137	Ö 153	⌐ 169	╣ 185	╔ 201	┘ 217	Θ 233	· 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	è 138	Ü 154	¬ 170	║ 186	╩ 202	┌ 218	Ω 234	· 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	ï 139	¢ 155	½ 171	╗ 187	╦ 203	█ 219	δ 235	√ 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	î 140	£ 156	¼ 172	╝ 188	╠ 204	▄ 220	∞ 236	n 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	ì 141	¥ 157	¡ 173	╜ 189	═ 205	▌ 221	φ 237	² 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	Ä 142	Pts 158	« 174	╛ 190	╬ 206	▐ 222	ε 238	■ 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	Å 143	ƒ 159	» 175	┐ 191	╧ 207	▀ 223	∩ 239	NBSP 255	0F

11 pav. IBM 437 koduotė

Antroje pusėje (8–F (80–F0) stulpeliai) 34 kodai (80–9A (128–154), 9F–A5 (159–165)), skirti Vakarų Europos tautų kalbų raidėms. Ši koduotės sritis vėliau buvo naudojama kitų kalbų grupių raidėms koduotėse, sudarytose šios koduotės pagrindu.

33 kodai (B0–DF (176–223), FE (254)) skirti pseudografikos ženklams. Kol buvo vien tekstiniai monitoriai ir tekstiniai spausdintuvai, tai buvo patogi priemonė lentelių rėmams ir nesudėtingiems grafiniams piešiniams piešti.

Likusi koduotės dalis skirta įvairiems specialiesiems ženklams, daugiausia matematikos simboliams, taip pat ir matematikoje vartojamoms graikiškoms raidėms.

Šios koduotės pagrindu buvo sudaryta nemažai koduočių kitoms kalbų grupėms. Dabar jos jau nebeaktualios ir jų nenagrinėsime, išskyrus turinčias lietuvių kalbos abėcėlės raides.

DOS sistemos koduotės ir lietuvių kalbos raidės

Pirmoji greitai paplitusių IBM PC genties kompiuterių operacinė sistema buvo vadinamoji DOS – diskinė operacinė sistema. Tai buvo pirmoji operacinė sistema, skatinusi kurti bei norminti koduotes, turinčias lietuviškus rašto ženklus. Buvo sukurtos kelios skirtingos koduotės. Aptarsime jas.

770 koduotė. 1989 metais buvo priimtas pirmasis Lietuvos koduočių standartas LST 1095-89 – koduotė 770, dar vadinama „IBM Baltic“ (12 pav.). Standartą parengė Virginijus Dadurkevčius, Danielius Ralys, Julius Sruogis ir Rimas Voldemaras, konsultuodamiesi su latviais ir estais. Pateiksime principus, kuriais buvo vadovautasi, kuriant šį standartą (Ralys, Dadurkevičius, 1991).

1. Koduote turi būti galima naudotis kuo didesnėje teritorijoje. Vienintelė teritorija, kurioje tuo metu dar nebuvo sunorminti nacionalinių ženklų kodai, buvo Baltijos valstybės, todėl manyta, jog šiuo standartu naudosis Lietuva, Latvija ir Estija.
2. Koduotėje, lyginant su IBM 437 koduote, tose pačiose pozicijose turi išlikti tie ženklai, kurie dažnai naudojami programuojant pseudografiką arba populiarūs spausdinant dalykinius raštus ar mokslinius straipsnius.
3. Lietuviškas, latviškas ar estiškas tekstas turi būti nesunkiai skaitomas, kai naudojamasi ir neperdirbtais spausdintuvais ar monitoriais, suderintais su IBM 437 koduote – vadinasi, Baltijos šalių raidės, kiek įmanoma, turi būti ten, kur IBM 437 koduotėje yra panašūs ženklai.

4. Pasauliniame moksle ir technikoje vis tvirtesnes pozicijas užima Vokietija. Įvertinus perspektyvas, tikslinga neliesti vokiečių kalbos ženklų, esančių 9 paveikslo 8 ir 9 (pažymėtuose 80 ir 90) stulpeliuose, juo labiau, kad dalis šių ženklų naudojama ir estų kalboje.

Visi pirmosios lentelės pusės (0–127 kodai) ženklai sutampa su ASCII koduote (žr. 9 pav.).

Antrosios lentelės pusės pagrindu buvo pasirinkta 437 koduotė. Lietuviškoms, latviškoms ir estiškoms raidėms buvo skiriamos vietos, kur buvo kitų kalbų savitosios (t. y. nesančios pagrindinėje lotynų abėcėlėje), išskyrus vokiečių, raidės. Dėl to koduotėje išliko specialieji 437 koduotės ženklai, taigi ir galimybė korektiškai dirbti

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	Č 128	Ė 144	Ā 160	▒ 176	└ 192	╨ 208	α 224	≡ 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	ü 129	ž 145	Ī 161	▒ 177	┴ 193	╤ 209	ß 225	± 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	ė 130	Ž 146	ķ 162	▓ 178	┬ 194	╥ 210	Γ 226	≤ 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	ā 131	õ 147	Ķ 163	│ 179	├ 195	╙ 211	π 227	≤ 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	ä 132	ö 148	ñ 164	┤ 180	─ 196	╘ 212	Σ 228	⌠ 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	ą 133	Õ 149	Ñ 165	╡ 181	┼ 197	╒ 213	σ 229	⌡ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	ļ 134	ū 150	Ū 166	╢ 182	╞ 198	╓ 214	µ 230	÷ 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	č 135	ų 151	Ų 167	╖ 183	╟ 199	╫ 215	τ 231	≈ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	ē 136	ģ 152	Ģ 168	╕ 184	╚ 200	╪ 216	Φ 232	° 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	Ē 137	Ö 153	⌐ 169	╣ 185	╔ 201	┘ 217	Θ 233	• 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	ę 138	Ü 154	¬ 170	║ 186	╩ 202	┌ 218	Ω 234	· 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	Ę 139	¢ 155	½ 171	╗ 187	╦ 203	█ 219	δ 235	√ 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	ī 140	Ļ 156	¼ 172	╝ 188	╠ 204	▄ 220	∞ 236	n 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	į 141	„ 157	Į 173	╜ 189	═ 205	▌ 221	φ 237	² 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	Ä 142	š 158	« 174	╛ 190	╬ 206	▐ 222	ε 238	■ 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	Ą 143	Š 159	» 175	┐ 191	╧ 207	▀ 223	∩ 239	NBSP 255	0F

12 pav. 770 koduotė

su programomis, naudojančiomis tuos ženklus. O tai labai svarbu, kadangi užsieninėse programose specialieji ženklai vienaip ar kitaip panaudojami ir interpretuojami pagal jų 437 koduotėje numatytą prasmę. Todėl nesklandumų naudojant užsieninę programinę įrangą (operacinių sistemų apvalkalus, tekstų rengykles, pašto programas ir pan.) pasitaikydavo retai.

Esame pastebėję tik du konfliktinius atvejus: 1) laiškuose, persiunčiamuose „Fidonet“ tinklu, dingdavo raidė į, 2) jeigu raidė Ą buvo pavartota aplanko varde, tai tokio aplanko negalėjo atverti „Norton Commander“ programa. Abu atvejus reikia laikyti minėtų programų klaidomis. Antroji klaida buvo ištaisyta „Norton Commander“ penktojoje versijoje. Pirmoji matyt ir liko neištaisyta, nes laiškus pradėjus siųsti internetu „Fidonet“ tinklo programinė įranga tapo nebeaktuali.

Nepanašu, kad „Norton Commander“ programos autoriai išnagrinėjo lietuvišką koduotę ir todėl ištaisė klaidą savo programoje. Labiau tikėtina, kad klaidą pastebėjo ir paprašė ištaisyti kas nors iš kitos valstybės, nes raidė Ą patenka į savitosioms raidėms skirtą vietą, kur tas pats kodas analogišku tikslu naudojamas ir kitose kalbose.

Pastebėsime, kad naujas ženklas yra ir lietuviškos atidaromosios kabutės (kodas 9D (157)). Kadangi tų laikų adatinių spausdintuvų skiriamoji geba buvo nedidelė, tai uždaromųjų kabučių funkcijas galėjo atlikti ASCII koduotės paprastosios (stačiosios) kabutės (kodas 22 (34)).

Dėl ženklų stygiaus nevienareikšmiškai buvo interpretuojami ir kai kurie kiti 437 koduotės specialieji ženklai. Pavyzdžiui, E1 (225) pozicijoje esančią raidę β matematikai laikė graikiška beta, o vokiečiai raide ß (es-cet), 237 pozicijoje esančią graikišką φ matematikai vartodavo ir tuščiai aibei arba diametrui žymėti, o norvegai vietoj raidės ø, nes jiems tik šios raidės ir trūko 437 koduotėje.

770 koduotė buvo pirmasis ir profesionaliai sukurtas lietuviškų rašmenų kodavimo standartas. Šio standarto rengimą autoriai yra aprašę ataskaitoje „Lietuviški rašmenys kompiuteryje“, kuri, deja, nebuvo publikuota.

771 koduotė. Maždaug tuo pačiu metu Vidmantas Balčytis parengė 771 koduotę, dar vadinamą „Kbl“. Faktiškai tai buvo rusiška-lietuviška koduotė, nes joje buvo ir rusų kalbos raidės, be to, joms buvo teikiama pirmenybė – paliktos tose pačiose vietose kaip ir Rusijoje naudojamoje kirilicos 866 koduotėje. Rusiškų raidžių yra 32. Jos užėmė visas vietas, kurias įprasta skirti raidėms. Savitosios lietuvių kalbos raidės buvo išdėstytos 220–223 ir 240–253 pozicijose vietoj 437 koduotės pseudografikos ir matematikos ženklų (žr. 11 pav.).

Dėl tokio lietuviškų raidžių išdėstymo kompiuterio naudotojas susidurdavo su daugybe nesklandumų. Paminėsime keletą jų.

1. Užsieninėse programose vartojami 9 matematikos ženklai ± ÷ ≥ ≤ ≈ ≡ √ ^{2} ^{n} virsdavo lietuvių kalbos raidėmis.
2. Programose nebuvo galima vartoti kai kurių pseudografikos ženklų, o dirbant su užsieninėmis programomis tokie ženklai virsdavo lietuvių kalbos abėcėlės raidėmis. Tais laikais pseudografikos ženklai buvo dažnai vartojami. Tai buvo vienintelė priemonė lentelėms įrėminti ir nesudėtingiems pseudografikos piešinukams formuoti, nes tikros grafikos kompiuteriuose dar nebuvo.
3. Tuo metu populiarioje tekstų rengyklėje „Word 5“ faktiškai nebuvo galima pasinaudoti visų ženklų rodymo veiksena, kurioje vietoj tarpus vaizduojančių taškų per vidurį „·“ buvo rodomos raidės Ū. Todėl, pavyzdžiui, vietoj eilutės: MŪSŲ·IR·JŪSŲ·ŽEMĖ buvo matoma eilutė: MŪSŲŪIRŪJŪSŲŪŽEMĖ.
4. Pseudografikos ženklams virstant raidėmis iškreipiamas tekstų konvertavimas. Pavyzdžiui, kai tekstų rengyklė „Word 5“ dokumentą rašo į tekstinį failą, išlaikydama teksto skirstymą į eilutes, raidės ą, č, Ą, Č virsta pliuso ženklu +.
5. Tais laikais elektroniniams laiškams persiųsti Lietuvoje buvo naudojamas UUCP protokolas ir vienintelė šiam protokolui tinkanti elektroninio pašto programa „Demos Mail“. Ši programa buvo sukurta Rusijoje ir skirta rusų kalba parašytiems laiškams persiųsti. Tačiau ji gerai persiųsdavo ir laiškus, parašytus lietuvių kalba, su sąlyga, jeigu jie koduoti 770 koduote. Laiškuose, koduotuose 771 koduote, savitąsias lietuvių kalbos raides ši programa pakeisdavo kitokiais ženklais. Situacija keistoka: 771 koduotė turi rusiškas raides (ir tai buvo viena jos plitimo Lietuvoje priežasčių), bet su ja koduotų laiškų „nesupranta“ rusiška pašto programa. Paaiškinimas paprastas: šioje koduotėje savitosios lietuvių kalbos abėcėlės raidės koduojamos kodais, kuriais nebuvo priimta koduoti raidžių (raidėms skirtas vietas buvo užėmusios rusiškos raidės).

Nepaisant trūkumų ir jų sukeltų nesklandumų, 771 koduotė Lietuvoje buvo labiausiai paplitusi, kompiuteriais prekiaujančios įmonės ją dėjo beveik į visus kompiuterius. O tai darė neigiamą įtaką lietuvių kalbos rašto ženklų vartojimui kompiuteriniuose dokumentuose, ypač elektroniniame pašte.

772 koduotė. Siekiant padėtį bent šiek tiek ištaisyti, buvo priimtas Lietuvos standartas LST 1284:1993, įteisinantis naują koduotę. Ji buvo taip pat rusiška-lietuviška, tiktai visos savitosios lietuvių kalbos raidės išdėstytos vietoj pseudografikos ženklų (181–184, 189, 190, 198, 199, 207–216 pozicijose).

Šioje koduotėje yra abiejų lietuviškų kabučių ženklai „ ir “. Paaukojus vien tik skirtingo storio linijų sankirtų pseudografikos ženklus, išliko kiti, svarbesni, pseudografikos ir matematikos ženklai.

774 koduotė. Lietuvių kalbos raidžių kodai yra tokie patys kaip ir 772 koduotėje. Tiktai vietoj rusiškų raidžių pateiktas įvairių Europos kalbų raidžių rinkinys. Problemos su pseudografikos ženklais išliko tos pačios, kaip ir 772 koduotėje. Šią koduotę apibrėžia Lietuvos standartas LST 1283.

775 koduotė. Lietuvių kalbos abėcėlės raidžių kodai sutampa su 772 koduotės kodais. Tiktai vietoj rusiškų raidžių įdėti kitų Baltijos valstybių raidžių kodai (13 pav.).

Tai buvo pirmoji (ir vienintelė) DOS sistemos koduotė su lietuviškomis raidėmis, kurią platino „Microsoft“ kompanija. Ji pirmą kartą buvo pateikta operacinėje sistemoje MS DOS, veikiančioje „Windows 95“ operacinėje sistemoje.

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	Ć 128	É 144	Ā 160	▒ 176	└ 192	ą 208	Ó 224	SHY 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	ü 129	æ 145	Ī 161	▒ 177	┴ 193	č 209	ß 225	± 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	é 130	Æ 146	ó 162	▓ 178	┬ 194	ę 210	Ō 226	“ 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	ā 131	ō 147	Ż 163	│ 179	├ 195	ė 211	Ń 227	¾ 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	ä 132	ö 148	ż 164	┤ 180	─ 196	į 212	õ 228	¶ 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	ģ 133	Ģ 149	ź 165	Ą 181	┼ 197	š 213	Õ 229	§ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	å 134	¢ 150	” 166	Č 182	Ų 198	ų 214	µ 230	÷ 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	ć 135	Ś 151	¦ 167	Ę 183	Ū 199	ū 215	ń 231	„ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	ł 136	ś 152	© 168	Ė 184	╚ 200	ž 216	Ķ 232	° 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	ē 137	Ö 153	® 169	╣ 185	╔ 201	┘ 217	ķ 233	· 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	Ŗ 138	Ü 154	¬ 170	║ 186	╩ 202	┌ 218	Ļ 234	· 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	ŗ 139	ø 155	½ 171	╗ 187	╦ 203	▉ 219	ļ 235	¹ 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	ī 140	£ 156	¼ 172	╝ 188	╠ 204	▄ 220	ņ 236	³ 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	Ź 141	Ø 157	Ł 173	Į 189	═ 205	▌ 221	Ē 237	² 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	Ä 142	× 158	« 174	Š 190	╬ 206	▐ 222	Ņ 238	■ 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	Å 143	¤ 159	» 175	┐ 191	Ž 207	▀ 223	’ 239	NBSP 255	0F

13 pav. 775 koduotė

Ši koduotė turėjo tokius pat trūkumus, susijusius su pseudografikos ženklais, kaip ir 772 ar 774 koduotės. Jos privalumas – latviškos ir estiškos raidės.

Šių kalbų raides turi ir „Windows“ sistemos koduotė 1257. Kadangi abi koduotės (775 ir 1257) yra daugiausia bendraujančiose terpėse (DOS ir „Windows“), tai logiška, kad jų abiejų ženklų aibės turi būti kuo artimesnės viena kitai.

DOS operacinė sistema, būdama „Windows“ terpėje, vaidino tik pagalbinį vaidmenį, todėl koduotės 775 trūkumai darėsi mažiau pastebimi. Taip ji tapo standartu de facto, o oficialiai įteisinta Lietuvos standartu LST 1590-1, pakeičiančiu visus ankstesnius DOS sistemos koduočių 770, 772 ir 774 standartus (LST 1095-89, LST 1284:1993, LST 1283:1993), tik 2000 metais.

DOS sistema nebenaudojama, tačiau jos koduotės dar išliko „Windows“ sistemos viduje failų (ir aplankų, nes sistemos viduje failais vaizduojami ir aplankai) vardams koduoti. Pereinamuoju laikotarpiu iš DOS į „Windows“ sistemą buvo svarbu, kad failų vardai abiejose sistemose būtų matomi vienodai. Todėl „Windows“ sistemoje jie užkoduojami dvejopai. Toks kodavimas išliko iki šiol. Supainiojus dvejopai koduotus failų vardus dar pasitaiko internacionalizacijos klaidų, kai tie vardai įtraukiami į duomenis, pavyzdžiui, į aplankų pakus pakavimo programose.

Koduočių įvairovė ir jų įtaka lietuvių kalbai. Toje pačioje operacinėje sistemoje ir tai pačiai kalbai naudoti kelias skirtingas koduotes nėra pagrindo. Bet DOS sistemos laikais taip ir nepavyko susitarti dėl vienos koduotės. Tai kėlė nepatogumų keičiantis tekstais. Jeigu teksto siuntėjo ir gavėjo kompiuteriuose nustatytos skirtingos koduotės, tai gavėjas turėjo arba perkoduoti tekstą į savo kompiuterio koduotę, arba pakeisti kompiuterio koduotę. Šiam tikslui buvo parengtos perkodavimo programos. Daugiausia žinomos buvo dvi: „BacoPage“ ir LiR.

„BacoPage“ programa DOS operacinę sistemą papildydavo lietuvių kalbos nuostatomis (į šią sistemą lietuvių kalbos lokalės parametrai nebuvo įtraukti): koduote, didžiųjų raidžių atitikimu mažosioms, raidžių rikiavimu ir kt. Paleidus šią programą operacinėje sistemoje atsirasdavo lietuvių kalba su tokiomis pat „teisėmis“, kaip ir kitos kalbos (pvz., prancūzų, vokiečių), kurios buvo įtrauktos į operacinės sistemos platinamąjį paketą. Toliau su lietuvių kalba tvarkydavosi pati sistema.

Programa operavo DOS sistemos parametrais. Todėl ji buvo priklausoma nuo sistemos konkrečios laidos. Atsiradus naujai sistemos laidai tekdavo išleisti ir naują „BacoPage“ programos laidą. Programos autorius tai padarydavo, bet kompiuterio naudotojui tai sudarė tam tikrų nepatogumų, nes, pakeitus sistemą naujesne, reikėjo gauti ir naują kalbos programos versiją. Be to, programa buvo nuosavybinė (ją parengė bendrovė „Baltic Amadeus“), už jos licenciją reikėjo mokėti. Tai buvo priežastys,

stabdančios jos naudojimą. LiR programa atliko tarpininko vaidmenį tarp operacinės sistemos ir jos išorinių teksto įvedimo (klaviatūros) bei išvedimo (monitoriaus ekrano, spausdintuvo) įrenginių vienaip ar kitaip, pagal žmogaus pasirinkimą, interpretuodama ženklų kodus. Operacinės sistemos nuostatų ji nekeitė, todėl buvo mažiau priklausoma nuo operacinės sistemos. Vienodai gerai tiko visoms operacinės sistemos laidoms, buvo platinama nemokamai. Toks jos paprastumas daugelį žavėjo ir ji buvo kur kas populiaresnė negu „BacoPage“ programa.

Kita vertus, kadangi LiR programa nekeitė operacinės sistemos nuostatų, tai sistema „nežinojo“, kuri koduotė parinkta ir laikė, kad joje veikia 437 koduotė ir pagal ją nustatydavo didžiųjų ir mažųjų raidžių atitikimą. DOS sistemoje failų ir aplankų varduose didžiosios raidės neskiriamos nuo mažųjų. Failų vardai buvo rašomi mažosiomis raidėmis, o aplankų – didžiosiomis, nežiūrint kuriomis raidėmis jie buvo renkami. Dėl to kildavo konfliktų dėl skirtingo jų varduose esančių raidžių kodų interpretavimo: sistemos viduje pagal 437 koduotę, išorėje (paprastai monitoriaus ekrane) – pagal parinktąją lietuvišką. Todėl naudojantis LiR programa failų ir aplankų varduose nebuvo galima vartoti savitųjų lietuvių kalbos abėcėlės raidžių. Vėliau, pradedant „Windows 95“ sistema, šios problemos nebeliko. Dar daugiau – DOS sistemoje sukurtų failų ir aplankų vardai buvo teisingai rodomi „Windows“ sistemoje ir atvirkščiai. Bet daugelio žmonių įprotis čia jų vengti išliko iki šiol.

Kirčiuotų raidžių koduotės. Savitųjų lietuvių kalbos abėcėlės raidžių yra 18 (9 didžiosios ir 9 mažosios), kirčiuotų raidžių – 68 (34 didžiosios ir 34 mažosios). Iš viso 86. Ir jos visos vartojamos drauge. Tai daug. Kaip jas visas sudėti į vieną koduotę?

Problema sprendžiama kurioje nors esamoje koduotėje, turinčioje pagrindinės lietuvių kalbos abėcėlės raides, kitus mažiau reikalingus ženklus (kitų kalbų raides, matematikos ir kai kuriuos kitus specialiuosius ženklus), esančius dešinėje kodų lentelės pusėje, pakeitus kirčiuotomis raidėmis. Paprastas lietuviškas tekstas (be kirčiuotų raidžių), koduotas tokia kirčiuotų raidžių koduote, nesiskirs nuo koduoto ta koduote, iš kurios padaryta kirčiuotų raidžių koduotė. Na, o tai, kad tokiame tekste nebus galima vartoti tų ženklų, kurių kodus užėmė kirčiuotos raidės, nebus didelė bėda, nes tekstų su kirčiuotomis raidėmis nėra daug.

Buvo parengtos koduotės, atitinkančios tris daugiausia naudojamas įprastas koduotes (770, 771, 775). Greta kirčiuotų raidžių į jas buvo įtraukta ir keletas reikalingesnių transkripcijos ženklų. Nė vienoje iš jų nebuvo labai retai vartojamų Ǯ ir ǯ raidžių su riestiniu kirčiu.

Nė viena iš šių koduočių nebuvo standartizuota. Tik 2000 metais, greičiau dėl koduočių standartų vientisumo, o ne dėl poreikio, buvo parengtas kirčiuotų raidžių

standartas LST 1590-2, atitinkantis tuo pat metu parengtą LST 1590-1 standartą (775 koduotę). Šio standarto koduotėje yra jau visos kirčiuotos raidės.

DOS sistema nebenaudojama. Todėl čia pateikta informacija apie šiai sistemai skirtas koduotes gali būti naudinga tik norintiems susipažinti su koduočių raida arba prireikus iškoduoti šiomis koduotėmis užrašytus tekstus.

ISO/IEC 8859 standartai ir „Linux“ sistemos koduotės

Dauguma Europos kalbų vartoja visas arba beveik visas pagrindinės lotynų abėcėlės raides (be Q, V, W, X ir kt.) ir keletą kitų lotynų raidžių (Æ, Þ, ß, Š ir kt.). Dažniausiai naujos raidės gaunamos prie pagrindinių lotynų abėcėlės raidžių pridėjus diakritinius ženklus. Tokių raidžių yra tiek daug, kad jos visos netelpa į vieną koduotę. Todėl atskiroms kalbų grupėms sudaromi ženklų rinkiniai, kurie vadinami „Lotynų 1“, „Lotynų 2“ ir t. t. Kiekvienam jų sudaroma atskira koduotė.

XX a. devintajame dešimtmetyje buvo pradėtas kurti tarptautinių standartų rinkinys ISO/IEC 8859. Tai bendras dviejų organizacijų ISO ir IEC (angl. *International Electrotechnical Commision* – Tarptautinė elektrotechnikos komisija) darbo rezultatas, todėl oficialiuose standartų pavadinimuose minimos abi organizacijos. Tačiau santrumpa IEC dažnai praleidžiama ir rašoma trumpiau: ISO 8859.

Tai labiausiai žinomas ir naudojamas standartų rinkinys. Pirmoji jo koduotė ISO 8859-1 (14 pav.) buvo sudaryta 1987 metais ir skirta Vakarų Europos tautų kalboms: airių, anglų, danų, islandų, ispanų, italų, norvegų, olandų, portugalų, suomių, švedų, vokiečių ir kt. Šių kalbų ženklų rinkinys buvo jau anksčiau sunormintas ir vadinamas „Lotynų 1“.

ASCII koduotėje pirmieji 32 kodai skirti valdymo kodams. Tarptautinė standartų organizacija nutarė analogiškai pasielgti ir su ISO 8859 koduočių pirmaisiais antrosios lentelės pusės kodais – juos palikti valdymo reikalams. Kita priežastis – palikti šiuos stulpelius tuščius, kad išliktų suderinamumas su sena įranga, ypač telekomunikacijų, pripažįstančia tik 7 bitų kodavimą. Kai į tokią įrangą patenka 8 bitų kodas, tai ji sugeba priimti tik pirmuosius 7 bitus. Aštuntasis bitas dingsta ir kodas virsta ASCII kodu, sumažėdamas dešimtainiu skaičiumi 128 (arba šešioliktainiu F0). Jeigu originalaus ženklo kodas yra iš intervalo 160–255, tai jis virsta kitu teksto ženklu. Duomenys sugadinami. Įrangai tokia klaida nekenkia: jai nesvarbu persiunčiamų tekstų turinys. Tačiau, jei teksto ženklas virstų valdymo kodu, tai būtų laikomas komanda. Savaime aišku, kad tokia komanda nereikalinga ir todėl gali sutrikdyti įrangos darbą.

Tais pačiais 1987 metais parengta antroji koduotė ISO 8859-2, skirta Vidurio Europos kalboms (čekų, kroatų, lenkų ir kt.). Ženklų rinkinys pavadintas „Lotynų 2“.

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	128	144	NBSP 160	° 176	À 192	Ð 208	à 224	ð 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	129	145	¡ 161	± 177	Á 193	Ñ 209	á 225	ñ 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	130	146	¢ 162	² 178	Â 194	Ò 210	â 226	ò 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	131	147	£ 163	³ 179	Ã 195	Ó 211	ã 227	ó 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	132	148	¤ 164	´ 180	Ä 196	Ô 212	ä 228	ô 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	133	149	¥ 165	µ 181	Å 197	Õ 213	å 229	õ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	134	150	¦ 166	¶ 182	Æ 198	Ö 214	æ 230	ö 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	135	151	§ 167	· 183	Ç 199	× 215	ç 231	÷ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	136	152	¨ 168	¸ 184	È 200	Ø 216	è 232	ø 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	137	153	© 169	¹ 185	É 201	Ù 217	é 233	ù 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	138	154	ª 170	º 186	Ê 202	Ú 218	ê 234	ú 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	139	155	« 171	» 187	Ë 203	Û 219	ë 235	û 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	140	156	¬ 172	¼ 188	Ì 204	Ü 220	ì 236	ü 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	141	157	SHY 173	½ 189	Í 205	Ý 221	í 237	ý 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	142	158	® 174	¾ 190	Î 206	Þ 222	î 238	þ 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	143	159	¯ 175	¿ 191	Ï 207	ß 223	ï 239	ÿ 255	0F

14 pav. Vakarų Europai skirta ISO 8859-1 koduotė

1988 metais buvo parengtos dar dvi koduotės kitoms Pietų Europos (trečioji koduotė) ir Šiaurės Europos (ketvirtoji koduotė) tautoms. 1987–1988 metais taip pat buvo parengtos koduotės arabų, graikų, hebrajų ir kirilicos rašmenis vartojančioms kalboms.

Atrodė, kad aštuonių bitų koduotės jau aprėpia visas lotyniškus rašmenis vartojančias kalbas. Tačiau ne visi jomis buvo patenkinti. Turkų rašmenys buvo įtraukti į ISO 8859-3 koduotę. Bet turkai buvo nepatenkinti, kad toje koduotėje per mažai Vakarų Europos kalbų rašmenų. Buvo parengta ISO 8859-9 koduotė, artima Vakarų Europai skirtai ISO 8859-1 koduotei – ji gauta iš jos, pakeitus šešias islandų kalbos raides turkų abėcėlės raidėmis.

Kurį laiką buvo manoma, kad devynių koduočių jau užteks. Kai kuriose programose koduotės numeriui nurodyti buvo skiriamas vienaženklio skaitmens laukas.

Naujų koduočių rengimas sustojo. Jų standartizavimas buvo atnaujintas tik 1998 metais. Vietoj vienos pernelyg ankštos ketvirtosios koduotės buvo parengtos dvi: Šiaurės tautų kalboms skirta ISO 8859-10 ir Baltijos tautoms ISO 8859-13. Taip pat buvo parengta ISO 8859-14 koduotė keltų grupės kalboms (bretonų, gėlų, velsiečių) ir ISO 8859-15 koduotė Vakarų Europos kalboms.

Penkioliktoji koduotė faktiškai yra pirmosios pakoregavimas: įtrauktos „pamirštos“ prancūzų kalbos raidės Œ ir Ÿ (tik didžioji, mažoji jau buvo), raidės Š ir Ž, reikalingos estams ir suomiams rusiškiems asmenvardžiams transkribuoti, euro ženklas €, šiems rašmenims skirti mažiau reikalingų ženklų (¤ ¦ ¨ ´ ¸ ¼ ½ ¾) kodai.

ISO 8859-16 koduotė yra mišri Vakarų ir Vidurio Europos kalboms tinkanti koduotė, turinti rumunų abėcėlės raides Ș ir Ț, kurios buvo „pamirštos“ ankstesnėse koduotėse (10 lentelė).

Koduočių tinkamumas įvairioms kalboms pateiktas 11 lentelėje. Tai pačiai kalbai tenka po kelias koduotes. Tačiau konkrečios kalbos terpėje vartojama tik kuri nors viena jų.

10 lentelė. ISO 8859 koduočių rinkinys

Koduotė	I leid. metai	Rašmenys	Regionas	Pastabos
ISO 8859-1	987	Lotynų 1	Vakarų Europa	Airijos gėlų, albanų, anglų, baskų, bretonų, danų, farerie čių, fryzų, galisų, grenlandų, islandų, ispanų, italų, katalonų, liuksemburgiečių, lotynų, norvegų, olandų, portugalų, (prancūzų), retoromanų, suomių, Škotijos gėlų, švedų, vokiečių
ISO 8859-2	1987	Lotynų 2	Vidurio Europa	Albanų, anglų, čekų, kroatų, lenkų, lotynų, (rumunų), slovakų, slovėnų, sorbų, vengrų, vokiečių
ISO 8859-3	1988	Lotynų 3	Pietų Europa	Anglų, esperanto, italų, lotynų, maltiečių, portugalų, (prancūzų), turkų, vokiečių
ISO 8859-4	1988	Lotynų 4	Šiaurės šalys	Anglų, danų, estų, grenlandų, latvių, lietuvių, lotynų, norvegų, (samių), slovėnų, suomių, švedų, vokiečių
ISO 8859-5	1988	Lotynų* ir kirilica		Baltarusių, bulgarų, makedoniečių, rusų, serbų, (ukrainiečių) ir kt.
ISO 8859-6	1987	Lotynų* ir arabų		
ISO 8859-7	1987	Lotynų* ir graikų	Graikija	Graikų

10 lentelė (tęsinys)

Koduotė	I leid. metai	Rašmenys	Regionas	Pastabos
ISO 8859-8	1987	Lotynų* ir hebrajų		
ISO 8859-9	1989	Lotynų 5	ISO 8859-1 modifikacija turkų kalbai	Airijos gėlų, albanų, anglų, baskų, bretonų, danų, fariečių, fryzų, galisų, grenlandų, ispanų, italų, katalonų, liuksemburgiečių, lotynų, norvegų, olandų, portugalų, retoromanų, (prancūzų), suomių, Škotijos gėlų, švedų, turkų, vokiečių
ISO 8859-10	1998	Lotynų 6	Šiaurės tautos (samiai, eskimai)	Airijos gėlų, anglų, danų, estų, fariečių, grenlandų, islandų, lietuvių, lotynų, norvegų, (samių), slovėnų, suomių, švedų, vokiečių
ISO 8859-11	1998	Lotynų*, tajų		
ISO 8859-12**		Lotynų*, devanagari		
ISO 8859-13	1998	Lotynų 7		Anglų, danų, estų, latvių, lenkų, lietuvių, lotynų, norvegų, slovėnų, suomių, švedų, vokiečių
ISO 8859-14	1998	Lotynų 8	Keltų grupės kalbos	Airijos gėlų, albanų, anglų, baskų, bretonų, danų, fryzų, galisų, grenlandų, ispanų, italų, katalonų, kornų, liuksemburgiečių, lotynų, menksiečių, norvegų, olandų, portugalų, (prancūzų), retoromanų, Škotijos gėlų, švedų, velsiečių, vokiečių
ISO 8859-15	1998	Lotynų 9	Vakarų Europa, įskaitant Prancūziją ir Estiją	Airijos gėlų, albanų, anglų, baskų, bretonų, danų, estų, fariečių, fryzų, galisų, grenlandų, islandų, ispanų, italų, katalonų, liuksemburgiečių, lotynų, norvegų, olandų, portugalų, prancūzų, retoromanų, suomių, Škotijos gėlų, švedų, vokiečių
ISO 8859-16	2001	Lotynų 10	Vidurio ir Rytų Europa	Airijos gėlų, albanų, anglų, italų, kroatų, lenkų, lotynų, rumunų, serbų, prancūzų, slovėnų, vengrų, vokiečių

* Tik tie lotynų rašmenys, kurie yra ASCII koduotėje.
** Šios koduotės standartas 1997 m. buvo panaikintas.

Pastaba. Lentelėje skliaustais pažymėtos kalbos, kurių ne visi rašmenys yra toje koduotėje, bet nesant kitos išeities, koduotė gali būti vartojama.

11 lentelė. Koduočių tinkamumas kalboms, vartojančioms lotyniškus rašmenis

Kalba	Koduotė ISO 8859-x									
	1	**2**	**3**	**4**	**9**	**10**	**13**	**14**	**15**	**16**
Airijos gėlų	1				9	10		14	15	16
Albanų	1	2			9			14	15	16
Anglų	1	2	3	4	9	10	13	14	15	16
Baskų	1				9			14	15	
Bretonų	1				9			14	15	
Čekų		2								
Danų	1			4	9	10	13	14	15	
Esperanto			3							
Estų				4		10	13		15	
Fareriečių	1				9	10			15	
Fryzų	1				9			14	15	
Galisų	1				9			14	15	
Grenlandų	1			4	9	10		14	15	
Islandų	1					10			15	
Ispanų	1				9			14	15	
Italų	1		3		9			14	15	16
Katalonų	1				9			14	15	
Kornų								14		
Kroatų		2								16
Latvių				4			13			
Lenkų		2			9	10	13			16
Lietuvių				4		10	13			
Liuksemburgiečių	1				9			14	15	
Lotynų	1	2	3	4	9	10	13	14	15	16
Maltiečių			3							
Menksiečių								14		
Norvegų	1			4	9	10	13	14	15	
Olandų	1				9			14	15	
Portugalų	1		3		9			14	15	
Prancūzų	(1)		(3)		(9)			(14)	15	16
Retoromanų	1				9			14	15	
Rumunų		(2)								16
Samių				(4)		(10)				
Serbų										16
Slovakų		2								
Slovėnų		2		4		10	13			16
Sorbų		2								
Suomių	1			4	9	10	13		15	
Škotijos gėlų	1				9			14	15	
Švedų	1			4	9	10	13	14	15	
Turkų			3		9					
Velsiečių								14		
Vengrų		2								16
Vokiečių	1	2	3	4	9	10	13	14	15	16

Pastaba. Jei koduotė kalbai tinka, tai lentelės langelyje įrašytas koduotės numeris. Gauname į eilutę surašytus visų koduočių, tinkančių kalbai, numerius. Koduočių, turinčių ne visus tos kalbos rašmenis, numeriai apskliausti.

Pirmoji ISO 8859 standartų grupės koduotė, turinti lietuvių kalbos abėcėlės raides, buvo ISO 8859-4 (15 pav.). 1989 metais buvo įteisinta ir kaip Lietuvos standartas LST 1093-89.

Šioje koduotėje buvo visos lietuvių kalbos abėcėlės raidės, bet mažoka specialiųjų ženklų, nes teko į vieną koduotę sudėti turtingas keleto tautų abėcėles. Todėl iš karto buvo imta projektuoti naują koduotę, atskirą Baltijos šalims. Juo labiau, kad į šią koduotę netilpo ir keletas samių (lapių) kalbų raidžių.

1992 metais bendromis Lietuvos ir Latvijos specialistų pastangomis buvo parengta „Baltic Rim“ koduotė. Ją projektavo Virginijus Dadurkevičius, Evaldas Kulbokas, Imants Mētra, Edmundas Mišeikis, Algimantas Oškinis, Danielius Ralys. Abi

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	128	144	NBSP 160	° 176	Ā 192	Đ 208	ā 224	đ 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	129	145	Ą 161	ą 177	Á 193	Ņ 209	á 225	ņ 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	130	146	ĸ 162	˛ 178	Â 194	Ō 210	â 226	ō 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	131	147	Ŗ 163	ŗ 179	Ã 195	Ķ 211	ã 227	ķ 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	132	148	¤ 164	′ 180	Ä 196	Ô 212	ä 228	ô 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	133	149	Ī 165	ī 181	Å 197	Õ 213	å 229	õ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	134	150	Ļ 166	ļ 182	Æ 198	Ö 214	æ 230	ö 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	135	151	§ 167	ˇ 183	Į 199	× 215	į 231	÷ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	136	152	¨ 168	¸ 184	Č 200	Ø 216	č 232	ø 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	137	153	Š 169	š 185	É 201	Ų 217	é 233	ų 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	138	154	Ē 170	ē 186	Ę 202	Ú 218	ę 234	ú 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	139	155	Ģ 171	ģ 187	Ė 203	Û 219	ė 235	û 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	140	156	Ŧ 172	ŧ 188	Ė 204	Ü 220	ė 236	ü 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	141	157	SHY 173	Ŋ 189	Í 205	Ũ 221	í 237	ũ 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	142	158	Ž 174	ž 190	Î 206	Ū 222	î 238	ū 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	143	159	¯ 175	ŋ 191	Ï 207	ß 223	ī 239	˙ 255	0F

15 pav. Šiaurės Skandinavijos ir Baltijos šalims skirta ISO 8859-4 koduotė

valstybės ją priėmė kaip nacionalinius standartus (LST 1282:1993 ir LVS 8-92) ir pateikė Tarptautinei standartų organizacijai ISO.

Priėmus naują Lietuvos standartą, ankstesnis LST 1093-89 neteko galios, bet tarptautinis naujos koduotės standartas ISO/IEC 8859-13 (16 pav.) pasirodė tik 1998 metais. 2000 metais šis standartas oficialiai įgijo ir Lietuvos standarto LST ISO/IEC 8859-13 statusą.

Šioje – tryliktoje ISO 8859 rinkinio – koduotėje yra visos mūsų kaimyninių tautų – danų, estų, latvių, lenkų, norvegų, suomių, švedų, vokiečių raidės (taip pat ir slovėnų). Todėl šiomis kalbomis galima laisvai naudotis lietuviškoje aštuonių bitų aplinkoje.

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
		SP	0	@	P	`	p			NBSP	°	Ą	Š	ą	š	00
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	
		!	1	A	Q	a	q			”	±	Į	Ń	į	ń	01
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241	
		"	2	B	R	b	r			¢	²	Ā	Ņ	ā	ņ	02
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242	
		#	3	C	S	c	s			£	³	Ć	Ó	ć	ó	03
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243	
		$	4	D	T	d	t			€	“	Ä	Ō	ä	ō	04
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244	
		%	5	E	U	e	u			„	µ	Å	Õ	å	õ	05
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245	
		&	6	F	V	f	v			¦	¶	Ę	Ö	ę	ö	06
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246	
		'	7	G	W	g	w			§	·	Ē	×	ē	÷	07
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247	
		(	8	H	X	h	x			Ø	ø	Č	Ų	č	ų	08
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248	
		)	9	I	Y	i	y			©	¹	É	Ł	é	ł	09
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249	
		*	:	J	Z	j	z			Ŗ	ŗ	Ź	Ś	ź	ś	0A
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250	
		+	;	K	[	k	{			«	»	Ė	Ū	ė	ū	0B
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251	
		,	<	L	\	l	|			¬	¼	Ģ	Ü	ģ	ü	0C
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252	
		-	=	M	]	m	}			SHY	½	Ķ	Ż	ķ	ż	0D
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253	
		.	>	N	^	n	~			®	¾	Ī	Ž	ī	ž	0E
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254	
		/	?	O	_	o				Æ	æ	Ļ	ß	ļ	’	0F
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255	

16 pav. Baltijos šalims skirta ISO 8859-13 koduotė

„Windows“ sistemos koduotės

Šios genties operacinėse sistemose naudojamos „Microsoft“ kompanijos koduotės, numeruojamos nuo 1250 iki 1257 ir sudarytos šiek tiek modifikavus ISO 8859 koduotes (12 lentelė).

12 lentelė. „Windows“ koduotės Europos kalbom

„Windows“ koduotė	ISO 8859 koduotė
Windows-1250	ISO 8859-2
Windows-1251	ISO 8859-5
Windows-1252	ISO 8859-1
Windows-1253	ISO 8859-7
Windows-1254	ISO 8859-9
Windows-1255	ISO 8859-8
Windows-1256	ISO 8859-6
Windows-1257	ISO 8859-13

„Windows“ koduotėse rašto ženklams panaudoti ir 8–9 stulpelių kodai, kurie ISO koduotėse buvo rezervuoti valdymo kodams. Čia įdėta įvairių kalbų kabutės ir kiti dažniau reikalingi ženklai.

Vakarų Europai sudaryta „Windows-1252“ koduotė. Nuo ISO 8859-1 koduotės ji skiriasi tik tuo, kad valdymo ženklams rezervuoti kodai (128–159) paskirti grafiniams ženklams (17 pav.), tarp jų ir prancūzų kalbos abėcėlės raidėms Œ, œ ir Ÿ, daugelyje kalbų vartojamoms raidėms Š ir š, kurių nebuvo ISO 8859-1 koduotėje.

Viena savitoji lietuvių kalbos raidė Š, esanti šioje koduotėje, naudos lietuvių kalbai neatnešė, bet sukėlė painiavos. Kai kurios programos, ją aptikusios lietuviškame tekste, palaiko Vakarų Europos kalboms priklausančia raide ir ją sudėtingai koduoja kaip svetimos koduotės raidę (pvz., kai kurios tinklalapių rengyklės ją užkoduoja pakaitos ženklų sekomis: Š → Š š → š – nors tai nėra klaida (ekrane matomos raidės Š ir š, o ne jas koduojantys ženklai), bet nelogiška koduotėje esančias raides keisti pakaitos sekomis ir sukelia nepatogumų tokį tinklalapį taisant grynojo teksto rašykle.

Ši koduotė beveik atitinka ir ISO 8859-15 koduotę. Čia matome ir trijų prancūzų kalbos abėcėlės raidžių kelią į kompiuterį: jų nebuvo ISO 8859-1 koduotėje, bet netrukus buvo įtrauktos į „Windows-1252“ koduotę, o vėliau ir į ISO 8859-15.

Vidurio Europos koduotė „Windows-1250“ daugiau skiriasi nuo savo prototipo ISO 8859-2 koduotės. Joje gana daug skirtumų lyginant su ISO 8859-3. Mat Vidurio

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
		SP	0	@	P	`	p	€		NBSP	°	À	Đ	à	ð	00
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	
		!	1	A	Q	a	q		‘	¡	±	Á	Ñ	á	ñ	01
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241	
		"	2	B	R	b	r	,	’	¢	²	Â	Ò	â	ò	02
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242	
		#	3	C	S	c	s	*ƒ*	“	£	³	Ã	Ó	ã	ó	03
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243	
		$	4	D	T	d	t	„	”	¤	‘	Ä	Ô	ä	ô	04
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244	
		%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ	05
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245	
		&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö	06
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246	
		'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷	07
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247	
		(	8	H	X	h	x	ˆ	˜	¨	¸	È	Ø	è	ø	08
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248	
		)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù	09
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249	
		*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú	0A
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250	
		+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û	0B
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251	
		,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü	0C
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252	
		-	=	M	]	m	}			SHY	½	Í	Ý	í	ý	0D
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253	
		.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ	0E
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254	
		/	?	O	_	o			ÿ	¯	¿	Ï	ß	ï	ÿ	0F
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255	

17 pav. Vakarų Europai skirta koduotė „Windows-1252“

Europos tautų abėcėlės gana turtingos ir čia įtraukta daugiau raidžių, nepatekusių į ISO 8859-3.

Lietuvai skirta 1257 koduotė (18 pav.). Be papildomų ženklų 80 ir 90 stulpeliuose ši koduotė nuo standartinės ISO 8859-13 skiriasi dar ir tuo, kad keturi ženklai yra kitose vietose:

„ (165 → 132, atidaromosios lietuviškos kabutės),

“ (180 → 147, uždaromosios lietuviškos kabutės),

” (161 → 148, kabutės – du kableliai viršuje),

’ (255 → 146, kabutės – vienas kablelis viršuje).

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	€ 128	144	NBSP 160	° 176	Ą 192	Š 208	ą 224	š 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	129	‘ 145	161	± 177	Į 193	Ń 209	į 225	ń 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	, 130	’ 146	¢ 162	² 178	Ā 194	Ņ 210	ā 226	ņ 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	131	“ 147	£ 163	³ 179	Ć 195	Ó 211	ć 227	ó 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	„ 132	” 148	¤ 164	180	Ä 196	Ō 212	ä 228	ō 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	… 133	• 149	165	µ 181	Å 197	Õ 213	å 229	õ 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	† 134	— 150	¦ 166	¶ 182	Ę 198	Ö 214	ę 230	ö 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	‡ 135	— 151	§ 167	· 183	Ē 199	× 215	ē 231	÷ 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	136	152	Ø 168	ø 184	Č 200	Ų 216	č 232	ų 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	‰ 137	™ 153	© 169	¹ 185	É 201	Ł 217	é 233	ł 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	138	154	Ŗ 170	ŗ 186	Ź 202	Ś 218	ź 234	ś 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	‹ 139	› 155	« 171	» 187	Ė 203	Ū 219	ė 235	ū 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	140	156	¬ 172	¼ 188	Ģ 204	Ü 220	ģ 236	ü 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	141	157	SHY 173	½ 189	Ķ 205	Ż 221	ķ 237	ż 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	142	158	® 174	¾ 190	Ī 206	Ž 222	ī 238	ž 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	143	159	Æ 175	æ 191	Ļ 207	ß 223	ļ 239	255	0F

18 pav. Baltijos šalims skirta 1257 koduotė

Skirtumų tarp ISO 8859-13 ir Windows-1257 koduočių atsirado dėl to, kad naujus, į rezervuotas vietas įdėtus specialiuosius ženklus visose „Windows“ terpės koduotėse norėta išlaikyti tose pačiose vietose. Tačiau dalis šių ženklų jau buvo koduotėje ISO 8859-13. Dviejose vietose tas pats ženklas negali būti, todėl tokie ženklai buvo perkelti į naujas pozicijas, esančias rezervuotų kodų (128–159) zonoje, o vietos, kurias jie buvo užėmę ISO 8859-13 koduotėje, liko tuščios.

Pastebėsime, kad skirtumų tarp ISO ir „Windows“ koduočių čia daugiau, negu tarp atitinkamų Vakarų Europos kalboms skirtų koduočių, bet mažiau, negu tarp Vidurio Europos kalboms skirtų koduočių.

Taigi pradėjus naudoti „Windows“ operacines sistemas, konkrečiau – nuo „Windows 95“, situacija su lietuviškomis koduotėmis pagerėjo. Išnyko DOS sistemoje buvusi įvairovė, pašaipiai vadinta „koduočių zooparku“. Tiesa, ir „Windows“ sistemai (iki „Windows 95“), kol jos platinamuosiuose paketuose dar nebuvo jokios lietuviškos koduotės, buvo sudaryta keletas mėgėjiškų koduočių. Tačiau, mūsų laimei, jos nespėjo paplisti.

„Mac OS“ sistemos koduotės

„Apple“ genties kompiuteriuose naudojamos savita „Mac OS“ operacinė sistema, taip pat ir koduotės. Šių koduočių pirmosios dalys sutampa su ASCII koduote, o antroje dalyje rašto ženklams koduoti išnaudojami visi 8 ir 9 stulpelių kodai nepaliekant jų valdymo reikalams. Dėl to į koduotes įtraukta daugiau ženklų, negu jų yra atitinkamose „Windows“ sistemų, juo labiau ISO 8859 grupės koduotėse. „Mac OS“ koduotės žymimos penkiaženkliais skaičiais:

10000 – Vakarų Europos;
10001 – japonų;
10002 – kinų tradicinė;
10003 – korėjiečių;
10004 – arabų;
10005 – hebrajų;
10006 – graikų;
10007 – kirilicos;
10008 – kinų supaprastintoji;
10010 – rumunų;
10017 – ukrainiečių;
10021 – tajų;
10029 – Vidurio Europos;
10079 – islandų;
10081 – turkų;
10082 – kroatų.

Vakarų Europos koduotė 10000 yra artima ISO 8859-1 koduotei.

Vidurio Europos koduotė 10029 yra artima ISO 8859-2 koduotei. Į ją įtrauktos visos savitosios lietuvių ir latvių kalbų abėcėlės raidės (19 pav.). Šioje koduotėje taip pat yra brūkšnys, lietuvių kalboje vartojamos kabutės, nemažai matematikos ženklų. Baltų kalbų abėcėles įtraukti į vieną koduotę kartu su Vidurio Europos kalbų abėcėlėmis pavyko dėl to, kad racionaliau parinkta specialiųjų ženklų aibė, išnaudoti visi antrosios kodų lentelės pusės kodai, rumunų kalbai skirta atskira koduotė.

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	Ä 128	ź 144	† 160	į 176	ņ 192	– 208	ŗ 224	ū 240	00
1	17	! 33	1 49	A 65	Q 81	a 97	q 113	Ā 129	Ď 145	° 161	Ī 177	Ń 193	— 209	Š 225	Ů 241	01
2	18	" 34	2 50	B 66	R 82	b 98	r 114	ā 130	í 146	Ę 162	≤ 178	¬ 194	“ 210	, 226	Ú 242	02
3	19	# 35	3 51	C 67	S 83	c 99	s 115	É 131	đ 147	£ 163	≥ 179	√ 195	” 211	„ 227	ů 243	03
4	20	$ 36	4 52	D 68	T 84	d 100	t 116	Ą 132	Ē 148	§ 164	ī 180	ń 196	‘ 212	š 228	Ű 244	04
5	21	% 37	5 53	E 69	U 85	e 101	u 117	Ö 133	ē 149	• 165	Ķ 181	Ň 197	’ 213	Ś 229	ű 245	05
6	22	& 38	6 54	F 70	V 86	f 102	v 118	Ü 134	Ė 150	¶ 166	∂ 182	Δ 198	÷ 214	ś 230	Ų 246	06
7	23	' 39	7 55	G 71	W 87	g 103	w 119	á 135	ó 151	ß 167	∑ 183	« 199	◊ 215	Á 231	ų 247	07
8	24	(40	8 56	H 72	X 88	h 104	x 120	ą 136	ė 152	® 168	ł 184	» 200	ō 216	Ť 232	Ý 248	08
9	25	) 41	9 57	I 73	Y 89	i 105	y 121	Č 137	ô 153	© 169	Ļ 185	… 201	Ŕ 217	ť 233	ý 249	09
10	26	* 42	: 58	J 74	Z 90	j 106	z 122	ä 138	ö 154	™ 170	ļ 186	NBSP 202	ŕ 218	Í 234	ķ 250	0A
11	27	+ 43	; 59	K 75	[91	k 107	{ 123	č 139	õ 155	ę 171	Ľ 187	ň 203	Ř 219	Ž 235	Ż 251	0B
12	28	, 44	< 60	L 76	\ 92	l 108	| 124	Ć 140	ú 156	¨ 172	ľ 188	Ő 204	< 220	ž 236	Ł 252	0C
13	29	- 45	= 61	M 77	] 93	m 109	} 125	ć 141	Ě 157	≠ 173	Ĺ 189	Õ 205	> 221	Ū 237	ż 253	0D
14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	é 142	ě 158	ģ 174	Í 190	ő 206	ř 222	Ó 238	Ģ 254	0E
15	31	/ 47	? 63	O 79	_ 95	o 111	127	Ź 143	ü 159	Į 175	Ņ 191	Ō 207	Ŗ 223	Ô 239	ˇ 255	0F

19 pav. Vidurio Europos ir Baltijos šalims skirta „Mac OS“ operacinės sistemos 10029 koduotė

3.3.2. Kalbų ir koduočių suderinamumas

Skirtingos koduotės atsirado dėl skirtingų ženklų aibių ir skirtingo ženklų kodavimo. Skirtingos ženklų aibės atsirado dėl to, kad pasaulio kalbose vartojamų rašto ženklų skaičius daug kartų viršija 8 bitų kombinacijų skaičių (256).

Skirtingi tų pačių ženklų kodai įvairiose koduotėse atsirado dėl susiklosčiusių tradicijų skirtingose operacinėse sistemose, kurios radosi ir tarpusavy konkuruodamos vystėsi maždaug tuo pačiu metu („Linux“ sistemos atsirado vėliau negu „Windows“ ir „Mac OS“, bet jose kodavimo principai ir pačios koduotės perimtos iš ankstesnių „Unix“ sistemų), o iš dalies dėl skirtingo tų pačių rašto ženklų vaidmens įvairiose kalbose.

Toje pačioje operacinėje sistemoje vienai kalbai pakanka vienos koduotės, išskyrus labai specialius atvejus. Toks specialus atvejis yra su lietuvių kalbos kirčiuotomis raidėmis. Visos lietuvių kalbos pagrindinės raidės ir visos kirčiuotos raidės telpa į vieną koduotę. Jos vienos ir pakaktų. Tačiau tokiu atveju nebelieka vietos kitų kalbų raidėms (latvių, lenkų, skandinavų), kurias norėtųsi turėti. Būtų neracionalu jas prarasti dėl kirčiuotų raidžių, kurios nedaug kam reikalingos. Todėl teoriškai galima laikyti, kad ženklų požiūriu lietuvių kalba su kirčiuotomis raidėmis prilygsta atskirai lietuvių kalbos atmainai, turinčiai savo koduotę.

Viena kalba vienoje valstybėje arba kitokioje etninėje teritorijoje faktiškai susiduria su trimis skirtingomis koduotėmis, priklausomomis nuo operacinių sistemų: ISO 8899-x, „Windows-x“ ir „Mac OS“. Tam, kad tekstas, perkeltas iš vienos operacinės sistemos į kitą būtų teisingai interpretuojamas, turi būti numatytas atitinkamas perkodavimas. Lietuviškiems tekstams būtinas perkodavimas tarp ISO 8859-13, „Windows-1257“ ir „Mac OS“ 10029 koduočių.

Ten, kur nuolat vyksta keitimasis tekstais, pageidautinas automatinis perkodavimas. Grynasis tekstas sudarytas vien iš ženklų, todėl vienareikšmiškai nustatyti jo koduotės nėra galimybės. Dėl to kartu su tekstu perduodama papildoma informacija apie jo koduotę.

Elektroninio laiško teksto koduotė užrašoma laiško antraštės lauke „Content type“, pavyzdžiui,

```
Content-Type: text/plain; charset=iso-8859-13
```

Paprastai pašto programa gautą laišką gali iškoduoti iš daugelio koduočių. Lietuvoje naudojamos pašto programos laiškus turi perkoduoti ir iš Lietuvoje nenaudojamų koduočių ISO-8859-4 ir ISO-8859-10. Išsiųsti originalų laišką pakanka kuria nors viena iš trijų minėtų koduočių. Geras tonas atsakymą į laišką išsiųsti gauto laiško koduote (atsakymo gavėjo programa su ta koduote tikrai susidoros).

Tinklalapio koduotė analogiškai nurodoma jo antraštėje gairės „meta“ atributu „charset“, pavyzdžiui,

```
<meta content="text/html; charset=windows-1257"
http-equiv="Content-type">
```

Apžvelgėme svarbesnes koduotes. Daugiau dėmesio skyrėme turinčioms lietuviškus rašto ženklus, ypač toms, kurios buvo daugiau naudojamos arba turėjo įtakos projektuojant kitas koduotes.

DOS sistemos koduočių įvairovė ir kai kurių jų trūkumai darė neigiamą įtaką lietuvių kalbos rašto ženklų vartojimui kompiuteriuose. Koduočių įvairovė išnyko

pradėjus visuotinai naudoti „Windows 95“ sistemą ir tolesnes „Windows“ laidas, nes kompiuterio naudotojas šiose sistemose rasdavo jų gamintojo įdėtą tik vieną šiai sistemai skirtą ir kvalifikuotai parengtą koduotę. Pasirinkimo nebuvo, bet jo ir nereikėjo: nėra argumentų, kad tame pačiame kompiuteryje, toje pačioje operacinėje sistemoje, tai pačiai kalbai reikėtų kelių skirtingų koduočių.

Šiuo metu daugiausia naudojama „Windows“ sistema, taigi ir jos koduotė „Windows-1257“. Ji įteisinta Lietuvos Respublikos standartu LST 1590-3.

Antroje vietoje – įvairios „Linux“ versijos. Jos naudoja ISO 8859 rinkinio koduotes. Lietuvių kalbos abėcėlė yra trijose šios serijos koduotėse: ISO 8859-4, ISO 8859-10 ir ISO 8859-13. Pastaroji vienintelė iš šių trijų įteisinta Lietuvos Respublikos standartu LST ISO/IEC 8859-13.

Visų lietuvių kalbos abėcėlės raidžių kodai ISO 8859-13 koduotėje sutampa su „Windows 1257“ koduotės kodais. Skiriasi tik kabučių kodai, o brūkšnio ISO 8859-13 koduotėje išvis nėra. Todėl teksto konvertavimas iš vienos koduotės į kitą reikalingas, bet ir nesant jo arba jam sutrikus tekstą perskaityti įmanoma – bus sugadintos tik kabutės ir brūkšnys.

3.4. Unikodas

Aštuonių bitų koduotės užtenka vienos arba kelių, bet ne visų Europos kalbų rašto ženklams. Tačiau jos neužtenka ir vienos Azijos kalbos hieroglifams. Išeitis – vienam ženklui skirti du arba daugiau baitų. Šiuo tikslu 1987 metais buvo pradėtas kurti unikodas (The Unicode Standard, 2006). Unikodą kuria ir toliau vysto specialiai šiam tikslui sukurtas konsorciumas „Unicode“, jungiantis stambias kompiuterijos firmas.

Unikode vienam ženklui skiriami du baitai, t. y. 16 bitų. Taigi jame yra 2^{16} = 65.536 kodų (čia įeina ir valdymo kodai, kurie negali būti skirti rašto ženklams).

1991 metais buvo paskelbta pirmoji unikodo laida.

Maždaug vienu metu su unikodu Tarptautinė standartų organizacija ISO pradėjo kurti 32 bitų koduotės ISO/IEC 10646 standartą. Koduotėje yra vietos 2^{32} ženklams. Toks ženklų skaičius sunkiai įsivaizduojamas ir jo turėtų užtekti ilgam, o gal ir visiems laikams.

1992 metais abu standartai buvo suderinti. Unikodas tapo ISO 10646 koduotės poaibiu. Tai reiškia, kad yra vieninga kodų erdvė nuo nulio iki šešioliktainio FFFFFFFF. Kodams nuo 0 iki FFFF pakanka dviejų 2 baitų arba 16 bitų, o didesniems – nuo 10000 iki FFFFFFFF – 4 baitų arba 32 bitų.

Nors kodų erdvė viena, tačiau tam, kad duomenų laikmenose būtų mažiau nereikšmingų nulių, koduojama vienu iš pasirinktų būdų: 16 bitų (vadinamu UTF-16) arba 32 bitais (vadinamu UTF-32 kodavimo būdu).

Ženklams kodai priskiriami palaipsniui, gerai apsvarsčius, nes laikomasi principo: jeigu kodas ženklui kartą priskirtas, tai jis nebegali būti atiduotas kitam ženklui, nors ankstesnis ženklas pasirodytų nebereikalingas. Pasiekti gerų rezultatų padeda suderinta „Unicode“ konsorciumo ir Tarptautinės standartų organizacijos ISO veikla. Konsorciume atstovaujamos kompiuterijos įmonės, todėl geriau matomi gamybiniai poreikiai. ISO atstovaujamos valstybės, todėl čia geriau matomi kalbų ir kultūrų poreikiai.

3.4.1. Kodavimas

Visą 16 bitų koduotę galima įsivaizduoti kaip 256 kodų lentelių (puslapių) rinkinį, kurių kiekvienoje yra po 256 kodus – tiek pat, kiek vienoje aštuonių bitų koduotėje. Tada pirmieji du šešioliktainiai kodo skaitmenys atitiks puslapio numerį, o kiti du – kodo poziciją puslapio lentelėje. Arba kaip vieną didelę lentelę, turinčią 256 stulpelius ir 256 eilutes.

Kodai numeruojami keturženkliais šešioliktainiais skaičiais nuo 0000 iki FFFF. Šiais numeriais, prieš juos prirašius U+, identifikuojami ženklai. Pavyzdžiui, raidės A kodas yra U+0041.

Pirmasis kodų puslapis (U+0000 – U+00FF) sutampa su ISO 8859-1 koduote. Tolesnius pustrečio puslapio užima kitos lotynų raidės, tarp jų ir savitosios lietuvių abėcėlės raidės. Toliau – kitų raštų rašmenys (graikų, kirilicos, armėnų, arabų ir kt.), įvairūs simboliai, matematikos ir kitokie ženklai.

Apie du trečdalius kodų užima kinų, japonų ir korėjiečių ideografiniai ženklai.

Dvi 16 bitų koduotės kodų sritys skirtos specialiems tikslams. Tai privačioji sritis ir surogatų sritis. Privačiosios srities kodai (U+E000 – U+F8FF) palikti atskirų žmonių grupių specifiniams poreikiams. Surogatų sritis (U+D8000 – U+DFFF) skirta 32 bitų kodams užrašyti 16 bitų kodų poromis ir taip padidinti ženklų, koduojamų 16 bitų koduote, skaičių.

Surogatų sritis perskirta į dvi dalis po 1024 kodus. Joje ženklai koduojami kodų poromis: vienas iš pirmosios dalies (U+D8000 – U+DBFF), kitas – iš antrosios (U+DC00 – U+DFFF). Taip kodų poromis galima užkoduoti 1024×1024 = 1.048.576 ženklų, arba užrašius dvejeto laipsniais $2^{10} \times 2^{10} = 2^{20}$. Tai 16 kartų daugiau, negu kodų 16 bitų koduotėje. Vadinasi, panaudojus surogatų poras ženklams, kurių kodai išeina už 16 bitų ribų, koduoti galima „įsiskverbti“ į 32 bitų srities dalį nuo 010000 iki 10FFFF.

Visas 000000–10FFFF intervalas pagal du pirmuosius šešioliktainius skaitmenis skirstomas į 17 dalių 00–10 (čia skaičiai šešioliktainiai) (13 lentelė), kurios vadinamos lentelėmis. Tai didelės lentelės, turinčios po 256 stulpelius ir 256 eilutes (iš viso 65.536 ženklų).

13 lentelė. 32 bitų koduočių lentelės

Lentelės Nr.	Rėžiai	Pavadinimas (paskirtis)
0	0000–FFFF	Pagrindinė daugiakalbė lentelė
1	10000–1FFFF	Papildoma daugiakalbė lentelė
2	20000–2FFFF	Papildoma ideografinių ženklų lentelė
3	30000–3FFFF	Dar viena papildoma ideografinių ženklų lentelė
4–13	40000–DFFFF	Dar niekam nepaskirta
14	E0000–EFFFF	Papildoma specialios paskirties lentelė
15	F0000–FFFFF	Papildoma privačiosios srities lentelė A
16	100000–10FFFF	Papildoma privačiosios srities lentelė B

Kol kas ISO 10646 koduotėje eksploatuojama tik tiek kodų, kiek galima užkoduoti 16 bitų unikodu, įskaitant ir surogatus, t. y. $2^{16} + 2^{20} = 65.536 + 1.048.576 =$ 1.114.112, kurių kodai yra intervale 000000–10FFFF. Taip koduojant nebetenka prasmės surogatų sritis su 2048 kodais, nes jos ženklai „išskleidžiami“ į 010000–10FFFF intervalo kodus, todėl ženklų skaičius šiek tiek mažesnis, negu turimų kodų: 1.114.112 – 2048 = 1.112.064.

ISO 10646 standarto dalis, apimanti 16 bitų kodus, vadinama pagrindine daugiakalbe lentele ir žymima ISO 10646-1. 2000 metais ji buvo įteisinta Lietuvos standartu LST ISO/IEC 10646-1.

Unikode ženklų daug, o kompiuterio klaviatūroje – tik labai maža jų dalis. Bet kurį unikodo ženklą galima surinkti jo kodu (žr. 3 priedą).

Lietuvių kalbos raidės unikode. Savitosios lietuvių kalbos abėcėlės raidės yra unikodo skyriuje, kuris vadinasi lotynų A plėtiniu, intervale U+0100 – U+017F (14 lentelė).

14 lentelė. Savitosios lietuvių kalbos raidės unikode

Didžiosios		Mažosios	
Ą	0104	**ą**	0105
Č	010C	**č**	010D
Ę	0118	**ę**	0119
Ė	0116	**ė**	0117
Į	012E	**į**	012F
Š	0160	**š**	0161
Ų	0172	**ų**	0173
Ū	016A	**ū**	016B
Ž	017D	**ž**	017E

Keleto dažniau vartojamų ženklų kodai pateikti 15 lentelėje.

15 lentelė. Kitų lietuvių kalbos rašto ženklų ir dažniau vartojamų matematikos ženklų kodai

Glifas	Kodas	Pavadinimas
„	201E	Atidaromosios kabutės
“	2013	Uždaromosios kabutės
–	2013	Brūkšnys
−	2212	Minusas
⩽	2264	Mažiau arba lygu (nedaugiau)
⩾	2265	Daugiau arba lygu (nemažiau)
≠	2260	Nelygu
√	221A	Kvadratinė šaknis
$\sqrt[3]{}$	221B	Kubinė šaknis
∧	2227	Konjunkcija
∨	2228	Disjunkcija
≡	2261	Ekvivalentumas
∏	220F	Sandauga
∑	2211	Suma
∈	2208	Priklauso aibei
∉	2209	Nepriklauso aibei
∅	2205	Tuščioji aibė
∩	2229	Aibių sankirta
∪	222A	Aibių sąjunga
∀	2200	Visuotinumo kvantorius
∃	2203	Egzistavimo kvantorius
∞	221E	Begalybė

3.4.2. Kompozicinės sekos

Unikode yra nulinio pločio ženklų grupė. Toks ženklas „užlipa“ ant prieš jį einančio normalaus ženklo (raidės) ir drauge su juo suformuoja naują ženklą (raidę). Taip galima gauti naujas raides su diakritiniais ženklais. Pavyzdžiui, po raidės P parašę nulinio pločio paukščiuko ženklą ˇ gausime raidę P̌, kurios unikode gali ir nebūti.

Taip ant vienos raidės galima uždėti ir kelis ženklus. Tokių ženklų kodai užrašomi išvardijant visų ženklo komponentų kodus ir juos skiriant pliuso ženklais, pavyzdžiui:

P̌: U+0050 + U+030C,

P̃̌: U+0050 + U+030C + U+0328 + U+0303.

Toks būdas naujoms raidėms gauti atėjo iš ankstesnių laikų, kai buvo vartojamas spausdinimo įtaisuose, kurie turėjo skurdžius šriftų rinkinius (rašomosiose mašinėlėse, pirmuose spausdintuvuose) – mechaniškai buvo spausdinami du ženklai į tą pačią vietą. Unikode visų kalbų abėcėlių raidės ir taip yra. Šis būdas skirtas retai vartojamiems ženklams, kurių unikode nėra, pavyzdžiui, fonetiniams tarmių ženklams, senovės raštuose vartotiems ženklams ir pan.

Nosinės raidės P ir dar su paukščiuku ir tilde unikode tikrai nėra. Bet gali atsitikti, kad tokiu būdu bus užkoduota ir kodą turinti raidė. Tai nebus klaida. Laikoma, kad abu kodavimai ekvivalentūs ir žymi vieną ir tą pačią raidę. Kai šitaip pridedami keli diakritiniai ženklai, tai jų eilės tvarka nesvarbi. Visa tai turi būti numatyta programose, operuojančiose ženklų ekvivalentumu, paieškos sistemose.

Unikode yra nulinio pločio ženklų grupė. Tam, kad matytųsi, ties kuria pagrindinio ženklo vieta jis dedamas, koduotėse kartu su ženklo glifu piešiamas punktyrinis apskritimas, nusakantis pagrindinio ženklo vietą (20 pav.).

20 pav. Keleto dažniau vartojamų diakritinių ženklų piešiniai ir jų kodai

3.4.3. Kirčiuotų raidžių kodavimas

Lietuvių kalba turi 68 kirčiuotas raides (34 didžiąsias ir 34 mažąsias). Jų įtraukimu į unikodą buvo susirūpinta per vėlai, apie 1999 metus (Aleknavičienė ir kt., 2005), po

to, kai buvo išleista antroji (1996 m.) ir baigiama rengti trečioji (2000 m.) unikodo laida, jo kūrėjai jau buvo sutvarkę raidėms skirtas kodų sritis ir nusprendę naujoms raidėms, kurias galima užkoduoti kompozicinėmis sekomis, atskirų kodų nebeskirti.

Dalis kirčiuotų raidžių sutampa su kitose kalbose vartojamomis įprastomis raidėmis, pavyzdžiui, raidė Ñ yra ispanų kalbos abėcėlėje, raidės Á, É, Í, Ó, Ú – čekų, islandų, ispanų, portugalų ir kitų kalbų abėcėlėse. Tokių raidžių yra 33, jos įtrauktos į unikodą ir turi jo kodus. Ta pati raidė lietuviams gali būti kirčiuota, o kurios nors kitos kalbos atstovui – pagrindinė jo kalbos abėcėlės raidė (unikode ir kitose koduotėse raidės tautybės neturi).

Kitos 35 kirčiuotas raidės atskirų kodų neturi. Jas tenka išreikšti kompozicinėmis sekomis. Kirčio ženklų kodai:

` U+0300
´ U+0301
~ U+0303

Kompozicinėmis sekomis (vienodumo dėlei) galima išreikšti ir tas raides, kurios turi atskirus kodus. Visų kirčiuotų raidžių kodai pateikti 16 lentelėje.

Raidė I gali turėti dešininį kirtį: Í í. Kirčio ženklas nenaikina taško ant mažosios i. Taip yra todėl, kad kirčiuota raidė nelaikoma atskira abėcėlės raidė – tai esama abėcėlės raidė, pažymėta kirčio ženklu. Todėl būtų nelogiška uždedant kirtį keisti pagrindinės raidės pavidalą. Kas kita, kai iš raidės i gaunama kita pagrindinės abėcėlės raidė, pavyzdžiui, latvių kalbos abėcėlėje esanti ī arba islandų í. Ją pakeisti pagrindine raide būtų klaida. Kirčio ženklas gali būti, gali ir nebūti – jis neprivalomas. Kirčiuojama tik ypatingais atvejais: žodynuose, lituanistinėje ir mokomojoje literatūroje, kartais ir įprastame tekste, kai nuo kirčio vietos priklauso žodžio prasmė.

Ši kirčiuotos i raidės savybė yra aprašyta ir unikode.

Analogiška situacija su kirčiuota j́. Tiktai į ją mažai kreipiama dėmesio, nes raidė j su kitais diakritiniais ženklais, išskyrus tašką, kitose kalbose retai vartojama. Lietuviška kirčiuota J taip pat labai reta. Ji buvo pamiršta įtraukti ir į pirmąsias kirčiuotų raidžių koduotes.

Dar ne visi šriftai gali suformuoti taisyklingus ženklų piešinius iš kompozicinių ženklų sekų. Todėl raidžių, kurias galima išreikšti tik kompozicinėmis sekomis, yra pateikti ir alternatyvūs kodai unikodo privačioje srityje. Čia bet kas gali pasirinkti kodus jam reikalingiems ženklams. Lyg ir patogu, bet nepraktiška: reikia pačiam pasirūpinti ir tų kodų realizacija: šriftais, įvedimu. Todėl į šį kodavimą reikia žiūrėti kaip į laikiną priemonę, kai nėra kitos išeities.

16 lentelė. Kirčiuotos raidės

Didžiosios			Mažosios		
Raidė	Unikodas		Raidė	Unikodas	
	Kodas	Kodų seka		Kodas	Kodų seka
À	00C0	0041 0300	**à**	00E0	0061 0300
Á	00C1	0041 0301	**á**	00E1	0061 0301
Ã	00C3	0041 0303	**ã**	00E3	0061 0303
Ą́		0104 0301	**ą́**		0105 0301
Ą̃		0104 0303	**ą̃**		0105 0303
È	00C8	0045 0300	**è**	00E8	0065 0300
É	00C9	0045 0301	**é**	00E9	0065 0301
Ẽ		0045 0303	**ẽ**		0065 0303
Ę́		0118 0301	**ę́**		0119 0301
Ę̃		0118 0303	**ę̃**		0119 0303
Ė́		0116 0301	**ė́**		0117 0301
Ė̃		0116 0303	**ė̃**		0117 0303
Ì	00CC	0049 0300	**ì**		0069 0307 0300
Í	00CD	0049 0301	**í**		0069 0307 0301
Ĩ	0128	0049 0303	**ĩ**		0069 0307 0303
Į́		012E 0301	**į́**		012F 0307 0301
Į̃		012E 0303	**į̃**		012F 0307 0303
Ý	00DD	0059 0301	**ý**	00FD	0079 0301
Ỹ		0059 0303	**ỹ**		0079 0303
J̃		004A 0303	**j̃**		006A 0307 0303
L̃		004C 0303	**l̃**		006C 0303
M̃		004D 0303	**m̃**		006D 0303
Ñ	00D1	004E 0303	**ñ**	00F1	006E 0303
Ò	00D2	004F 0300	**ò**	00F2	006F 0300
Ó	00D3	004F 0301	**ó**	00F3	006F 0301
Õ	00D5	004F 0303	**õ**	00F5	006F 0303
R̃		0052 0303	**r̃**		0072 0303
Ù	00D9	0055 0300	**ù**	00F9	0075 0300
Ú	00DA	0055 0301	**ú**	00FA	0075 0301
Ũ	0168	0055 0303	**ũ**	0169	0075 0303
Ų́		0172 0301	**ų́**		0173 0301
Ų̃		0172 0303	**ų̃**		0173 0303
Ú		016A 0301	**ú**		016B 0301
Ũ		016A 0303	**ũ**		016B 0303

3.4.4. Baitų eiliškumas

Spausdintame tekste baitų kodo baitų eilė savaime aiški: pirmiau aukštesnysis baitas, po jo žemesnysis. Taip, kaip ir skaičiaus skaitmenys. Tačiau kompiuteriuose skaičiai dažniausiai koduojami atvirkščiai. Mat su taip koduotais skaičiais paprasčiau atlikti aritmetines operacijas: tada jas galima atlikti nuo kodo pradžios, nes čia žemiausia skaičiaus skiltis. Todėl unikode įteisintas dvejopas baitų eiliškumas: *mažėjančių baitų* (kaip ir spausdintame tekste) ir *didėjančių baitų*. Didėjančių baitų eiliškumas yra kompiuteriuose, naudojančiuose „Intel“ procesorius, mažėjančių – HP, IBM, „Motorola 6800“ procesorius.

Baitų eiliškumui nustatyti vartojami du kodai: U+FEFF ir U+FFFE. Kodui U+FEFF priskirtas ženklas, vadinamas *nulinio pločio jungiamuoju tarpu*. Kodas U+FEFF, įrašytas teksto (failo) pradžioje, neturi jokios įtakos tekstui – tekstas su juo matomas lygiai taip pat, kaip ir be jo. Todėl šis kodas tinka baitų eiliškumui indikuoti. Kodui U+FFFE nepriskirtas joks ženklas, jis yra veidrodinis kodo U+FEFF atspindys (t. y. gaunamas iš kodo FEFF sukeitus vietomis baitus išreiškiančių raidžių poras).

Procesorius, perskaitęs pirmuosius du failo baitus, juos išdėsto jame priimtu eiliškumu (pirmas baitas, antras baitas arba atvirkščiai) ir iš jų suformuoja keturženklį šešioliktainį skaičių. Jeigu gaunamas skaičius FEFF, tai rodo, jog baitų eiliškumas faile toks pat, kai ir procesoriuje. Jeigu gaunamas skaičius FFFE, tai aišku, kad baitų eiliškumas faile yra priešingas negu procesoriuje ir failą reikia perkoduoti. Štai dėl to kodui FFFE nepriskirtas joks ženklas (jeigu jį atitiktų koks nors ženklas, tai procesorius negalėtų nustatyti, ar šį kodą reikia interpretuoti kaip ženklą, ar tai tik kodo FEFF atspindys).

3.5. Kodų pakaitalai

Koduočių galimybės augo didėjant bitų, skiriamų vienam ženklui, skaičiui: 5→6→7→8→16→32. Atsiradus naujai koduotei su didesniu bitų skaičiumi, atsiranda atgalinio suderinamumo su ankstesne koduote, turinčia mažiau bitų ir dar tebevartojama prieš tai sukurtoje įrangoje, poreikis. Tai ypač svarbu vykstant informacijos mainams.

Penkių ir šešių bitų koduotės kompiuteriuose buvo epizodinis reiškinys, nepalikęs pėdsakų ilgesniam laikui.

Kodavimo septyniais bitais, o konkrečiau – ASCII koduote, pradžia sutapo su interneto prototipo ARPANET tinklo ir paties interneto kūrimo pradžia. Visi šie darbai buvo intensyviai daromi JAV. Anglų kalbai pakako septynių bitų koduotės. Buvo pagaminta daug įrangos, ypač internetinės, pritaikytos septynių bitų koduotėms. Visa tai lėmė, kad septynbičio kodavimo palikimas jaučiamas ir dabar, praėjus pusei šimtmečio nuo jo atsiradimo.

3.5.1. Aštuonių bitų kodų pakaitalai

Siekiant prisiderinti prie senos telekomunikacijų įrangos buvo sukurta įvairių būdų aštuonių bitų kodams išreikšti septynių bitų kodais. Juos galima suskirstyti į dvi grupes.

1. Monolitinis teksto perkodavimas. Visas tekstas, kaip monolitinė bitų seka, paverčiamas šešiabičių kodų seka, kuri toliau paverčiama tais ASCII kodais, kuriuos gali persiųsti bet kuri septynbitė įranga. Tai nėra trivialus sekos skaidymas į gabaliukus po 6 bitus, nes reikia išvengti valdymo kodų, kuriuos telekomunikacijų sistemos palaikytų joms skirtomis komandomis, o ne duomenimis. Būdas ekonomiškas, bet iš koduotos sekos negalima išskirti, skaityti ir analizuoti atskirų ženklų neiškodavus viso teksto.

2. Individualusis ženklų perkodavimas. Tai aštuonbitės koduotės ženklų, kurių nėra ASCII koduotėje, kodavimas pakaitos sekomis, sudarytomis iš ženklų, esančių ASCII bitų koduotėje. Kiekvienas ženklas perkoduojamas individualiai (jeigu jį reikia perkoduoti). Todėl iš koduotos sekos galima išskirti, skaityti ir analizuoti atskirus ženklus, neiškodavus viso teksto. Kodavimas neekonomiškas, kai tekste yra daug ženklų, nesančių ASCII koduotėje.

Įvairūs kodavimo būdai ir jų naudojimo sritys pateikiamos 17 lentelėje.

Toliau aptarsime kodavimo būdus pagal jų taikymo sritis.

Elektroninis paštas. 1993 metais priimtame MIME standarte RFC 1521, kurį 1996 metais pakeitė naujas standarto leidimas RFC 2045, pateikti du kodavimo būdai B64 (B) ir QP (Q), skirti elektroniniam paštui ir turintys po du variantus. Variantai skiriasi tik tuo, kad juose numatytos šiek tiek skirtingos privalomų koduoti ženklų aibės, nes reikia koduoti ir kai kuriuos ženklus, esančius ASCII koduotėje, vartojamus tarnybiniams tikslams: vienus laiško antraštės laukuose, kitus laiško tekste ir prieduose. Išsamiau apie tai galima sužinoti RFC 2045 standarte.

17 lentelė. Kodavimo būdai. M – monolitinis, I – individualusis. Kabliataškiai pavyzdžių stulpelyje įeina į pakaitos sekas (t. y. nėra pavyzdžių sąrašo skyrybos ženklai)

Kodavimo būdas	**M/I**	**Naudojimo sritis**	**Pavyzdžiai**	**Pastabos**
B	M	El. laiškų antraštės		
B64, Base64	M	El. laiškų tekstai ir priedai		Nuo B skiriasi tik koduojamų ženklų aibe
Q, =FF	I	El. laiškų antraštės	Ą → =C0 č → =E8 = → =3D ačiū → a=E8i=FB	
QP, Quoted printable, =FF	I	El. laiškų tekstai ir priedai	ą → =E0 č → =E8 = → =3D 2×2=4 → 2=D72=3D4	Nuo Q skiriasi tik koduojamų ženklų aibe
Ženklo identifikatorius	I	Tinklalapiai	Ä → Ä ä → ä ÁbÄ → ÁbÄ þ → þ Š → Š & → &	
&#9..9;	I	Tinklalapiai	Ä → Ä ä → ä þ → î ÁbÄ → ÁbÄ Š → Š & → & 文 → 文	9...9 – dešimtainis ženklo kodas unikode
%FF	I	Failų ir aplankų vardai internete	ą → %E0 č → %F0 % → %25 ačiū → a%F0i%FB	Dabar retai naudojama; žr. UTF8 + %FF
UTF8 + %FF	I	Failų ir aplankų vardai internete	ą → %C4%85 č → %C4%8D % → %25 ačiū → a%C4%8Di%C5%AB 文 → %E6%96%87	
\uFFFF	I	Išteklių eilutėse	ė → \u0117 Þ → \u00DE	FFFF – ženklo kodas unikode

Perkoduojant QP būdu ženklai, nesantys ASCII koduotėje, pakeičiami pakaitos sekomis. Sekos pradžią rodo ženklas =. Toliau eina dviženklis šešioliktainis ženklo kodas. Tam, kad būtų galima atskirti tikrą ženklą =, žymintį lygybę (jis turi būti matomas tekste) nuo tokios pat ifvaizdos ženklo, atliekančio tarnybines funkcijas, sutarta tikrą lygybės ženklą išreikšti pakaitos seka =3D.

QP kodavimas priimtinas lotynų rašmenis vartojančioms kalboms, nes raidžių, kurias reikia koduoti, nėra daug. Todėl tekstą nors ir nepatogu, bet įmanoma perskaityti ir tada, kai dėl persiuntimo sutrikimų jis gaunamas neiškoduotas. Pavyzdžiui, žodis *Mėnulis* būtų užkoduotas taip: M=EBnulis.

Šis kodavimas neekonomiškas kitokius raštus vartojančioms kalboms (pvz., graikų, kirilicą), kurių kiekvieną raidę reikia koduoti. Tada persiunčiamas tekstas pailgėja beveik tris kartus (lieka neperkoduoti tik skaitmenys, skyrybos ženklai). Nors ženklai koduojami individualiai, naudos iš to mažai, nes neiškoduotą tekstą skaityti sunku – reikia iššifruoti kiekvieno ženklo pakaitos seką. Todėl tokioms kalboms naudotinas B64 kodavimas.

Ankstesnėse pašto programose kodavimo būdą reikėdavo parinkti laiško siuntėjui, šiuolaikinės programos pačios parenka kodavimo būdą ir įrašo jo pavadinimą į laiško antraštę, paanalizavusios išsiunčiamą tekstą:

```
Content-transfer-encoding: base64
Content-transfer-encoding: Quoted-printable
```

Programa, iškoduojanti gautą laišką, turi turėti informaciją, kokia koduote buvo parašytas laiškas. Ji įrašoma į šiam tikslui skirtą antraštės lauką, pavyzdžiui,

```
Content-type: text/plain; charset=windows-1257
Content-type: text/plain; charset=iso-8859-13
```

Laikoma, kad laiško antraštėje esanti informacija (laiško tema, ypač siuntėjo asmenvardis) yra labai svarbi, svarbesnė negu laiško tekste, ir jokiu būdu negali būti sugadinta dėl nesklandumų persiuntimo trakte. Todėl MIME standarte nustatyta, kad koduotė ir kodavimo būdas atskirai nurodomas kiekviename lauke to lauko tekstui, pavyzdžiui:

```
From: =?ISO-8859-13?Q?Jonas_Jon=EBnas?= jonas@xx.zz.lt
To: =?ISO-8859-13?Q?J=FBrat=EB_J=FBrait=EB?= jrt@zz.xx.lt
```

Čia ženklų grupės =? ir ?= išskiria koduotą tekstą – atlieka panašią funkciją, kaip skliaustai arba kabutės, o klaustukai frazės viduryje atskiria vieną nuo kito įrašo komponentus. Jeigu įraše būtų tikras klaustukas (pvz., temos lauke), tai jis būtų koduojamas pakaitos seka =3F.

Čia visur rašėme angliškus laiško antraščių laukų pavadinimus. Jie yra tarnybiniai žodžiai, vartojami telekomunikacijų įrangoje, keliauja su laišku, bet laiško skaitytojui nematomi. Skaitytojas savo kompiuteryje juos mato sava kalba: lietuvis lietuviškus, vokietis vokiškus – taip, kaip juos išvertė pašto programos lokalizuotojas.

Kaip matome, elektroniniame pašte kodavimo problemų buvo daug: reikėjo suderinti senąjį septynių bitų kodavimą su aštuonių bitų kodavimu. Tos problemos išspręstos ir laiškus siuntinėjame nejausdami, kad jų tekstai įvairiai perkoduojami. Tačiau pašto viduje dar tebėra 7 bitų įrangos komponentų, sudarančių galimybę programinės įrangos kūrėjams klysti. Todėl lokalizuotojas turi būti budrus ir prieš programos lokalizavimą nuodugniai patikrinti, ar joje nėra internacionalizacijos klaidų, susijusių su ženklų kodavimu.

Tinklalapiai. Seniausias metodas – raides, kurių nėra ASCII koduotėje, pakeisti specialiai šiam tikslui sugalvotais identifikatoriais – trumpais angliškais raidžių pavadinimais (pvz., þ → `thorn`, Þ → `THORN`) arba, jei raidė su diakritiniu ženklu, seka iš raidės be diakritinio ženklo ir angliško diakritinio ženklo pavadinimo, dažnai sutrumpinto (pvz., Á → `Aacute`, á → `aacute`). Pakaitos sekos pradžia žymima ženklu &. Kadangi sekų ilgiai nevienodi, tai jos pabaiga nurodoma kabliataškiu (þ → `&thorn;` Þ → `&THORN;` Á → `&Aacute;` á → `&aacute;`).

Tam, kad būtų galima tekste parašyti tikrą ženklą &, tenka ir jį koduoti: & → `&`.

Tikram kabliataškiui atskirti nuo tarnybinio pakaitos sekos ženklo nereikia – pirmasis sekoje sutiktas kabliataškis yra tarnybinis, visi kiti – tikrieji. Tam, kad ši taisyklė galiotų, reikia, kad kabliataškių nebūtų sekos viduje. Bet čia jie ir nebūtini.

Kol internetas buvo naudojamas tik didesnėse Vakarų Europos valstybėse, o koduotinų raidžių buvo nedaug, šis metodas neblogai tiko. Pradėjus vartoti aštuonių bitų koduotes saityne (žiniatinklyje), raidžių, esančių pasirinktoje tinklalapio koduotėje, kodavimas pakaitos sekomis pasidarė nebereikalingas. Jo prireikia tik norint parašyti raidę ar kitokį ženklą, kurio nėra toje koduotėje. Pavyzdžiui, jeigu lietuviškame tinklalapyje, kurio koduotė „Windows-1257“, norime parašyti raides Þ, ð.

Šis metodas dar ir dabar naudojamas tinklalapių tekste esantiems ženklams < ir > (tikriems) koduoti:

< → `<`
> → `>`

Mat šie ženklai vartojami kaip tarnybiniai hiperteksto gairėms žymėti.

Savitosioms lietuvių kalbos abėcėlės raidėms šis kodavimo būdas nebuvo naudojamas. Į naršykles neįtrauktos ir tokių raidžių pakaitos sekos (gal jos ir nebuvo pa-

rengtos). Bet išradingesni lietuviai ir latviai rado, kaip šiuo kodavimu pasinaudoti. Ilgą laiką interneto naršyklės neatpažindavo į tinklalapių antraštes įrašytų koduočių pavadinimų ir laikydavo, kad visi tinklalapiai parašyti ISO 8859-1 koduote. Pasinaudodami šia kodavimo spraga, tinklalapių autoriai lietuviškas savitąsias raides pakeisdavo pakaitos sekomis tų „vakarietiškų“ raidžių, kurių kodai ISO 8859-1 koduotėje sutampa su atitinkamų lietuviškų raidžių kodais „Windows-1257“ koduotėje. Pavyzdžiui, raidę Ą atitinka raidė À, kurios pakaitos seka À, raidę Į – raidė Á, kurios seka &Aaccute; ir t. t. Pakaitos sekas naršyklės paversdavo kodais, o kompiuteris kodus paversdavo raidėmis pagal lietuvišką koduotę („Windows-1257“).

Kartu su unikodu atsirado nauja galimybė raidėms identifikuoti pakaitos sekose – vietoj raidinio raidės identifikatoriaus vartoti jos kodą unikode. Į pakaitos seką rašomas dešimtainis kodas su ženklu # prieš jį. Pavyzdžiui, raidė Š koduojama taip: `&#352;`.

Šia galimybe pasinaudojama, kai į aštuoniais bitais koduotus tinklalapius reikia įrašyti vieną kitą toje koduotėje nesantį ženklą. Tai daroma analogiškai, kaip ir raidiniais ženklų identifikatoriais, tik paprasčiau ir sistemingiau: užuot sugalvojus pakaitos raidžių seką kiekvienam ženklui ir ją įtraukus į naršykles ir kitas interneto programas, galima pasinaudoti tvirtai paskirtais unikodo kodais. Ir pats užrašas, būdamas kalbiškai neutralus, geriau dera su internacionalizacijos idėja.

Dabar ir tarnybinių ženklų pakaitos sekas galima rašyti vartojant unikodo kodus:

```
& → &#38;
< → &#60;
> → &#62;
" → &#34;
```

Lokalizuojant programas arba tinklalapius senuoju būdu užrašytas pakaitos sekas reikėtų keisti unikodinėmis.

Senesnės tinklalapių rengyklės ir kitos darbo su tinklalapiais programos kai kuriuos skyrybos ženklus (dažniausiai brūkšnį ir kabutes) arba raides (dažniausiai Š) be reikalo keičia pakaitos sekomis. Šias programų vietas lokalizuojant reikėtų pakoreguoti. Jų gal nereikėtų vadinti internacionalizacijos klaidomis, nes neturi įtakos galutiniam produktui – tinklalapiui. Bet vadinti internacionalizavimo stiliaus klaidomis visai natūralu: kam vietoj vieno ženklo, atpažįstamo iš jo piešinio, rašyti kelių ženklų pakaitos seką, neturinčią vaizdinio ryšio su tuo ženklu.

Failų ir aplankų vardai. Kompiuteryje esančių failų ir aplankų varduose galima vartoti visas 8 bitų koduotėje esančias raides. Tačiau tinkluose, kai duomenys perduodami tarp kompiuterių, atsiranda problemų dėl skirtingų koduočių ir dėl pasenusios tinklų įrangos, naudojančios 7 bitų kodavimą.

Septynių bitų kodavimo problemos išsprendžiamos naudojant pakaitos sekas, prasidedančias procento ženklu su toliau einančiu dviženkliu šešioliktainiu ženklo kodu 8 bitų koduotėje.

Koduočių suderinamumui to nepakanka – nėra informacijos, kuri koduotė buvo panaudota tikriems failo vardo ženklams koduoti. Todėl toks kodavimo būdas galėjo būti naudojamas informacijos mainams tik tarp kalbų, vartojančių tą pačią koduotę.

Atsiradus unikodui atsirado galimybė išspręsti ir antrąją problemą – koduočių suderinamumą. Raidė, nesanti ASCII koduotėje, koduojama dvigubai: pirmiau UTF-8 kodu (koduočių suderinamumui), po to – %FF (suderinamumui su septynbite tinklų įranga). Kadangi ir UTF-8 yra ne tikra koduotė, o unikodo pakaitalas, tai išviso turime trigubą kodavimą: unikodas → UTF-8 → %FF. Apie UTF-8 kodavimą pakalbėsime kitame skyrelyje.

Taip koduojami laiškų priedų, persiunčiamų elektroniniu paštu, vardai, failų ir aplankų vardai, esantys tinklalapių adresuose (po pasvirojo brūkšnio /).

Laiško siuntėjas ir gavėjas priedų failų vardus mato tikrus, neperkoduotus. Koduotus juos galima pamatyti tik analizuojant pirminį laiško tekstą.

3.5.2. Unikodo kodavimas aštuoniais bitais (UTF-8)

Šuolis nuo 8 bitų prie 16 yra didesnis negu nuo 7 prie 8 bitų ne vien dėl to, kad kodas dvigubai ilgesnis, bet ir dėl to, kad kompiuteriai iki šiol tebėra aštuonbičiai (t. y. juose naudojamas baitų, turinčių 8 bitus, adresavimas). Pereinant nuo 7 prie 8 bitų tokios problemos nebuvo, nes kompiuteriai jau tada buvo aštuonbičiai, tik aštuntas bitas nebuvo panaudojamas. Reikėjo susiderinti tik su 7 bitų programine įranga, kuri tebėra ir dabar. Todėl sakome, kad šuolis prie 16 bitų ne „buvo“, o „yra“.

Siekiant prisiderinti prie 8 bitų, unikodo ženklų perkodavimas 8 bitais yra apibrėžtas pačiame unikode ir vadinamas UTF-8 kodavimu. Tuo jis skiriasi nuo įvairių 8 bitų ženklų kodavimo 7-iais bitais būdų, skirtingų skirtingose taikymo srityse. Todėl UTF-8 kodavimą galima laikyti universaliu. Kartais jis tapatinamas su pačiu unikodu.

Vienas 2 baitų unikodo kodas koduojamas 1, 2 arba 3 baitais, o kodų pora iš unikodo surogatų srities (4 baitai) – 4 baitais (18 lentelė).

18 lentelė. Unikodo ženklų kodavimas UTF-8 kodu

Unikodas (du baitai)	**UTF**				
	Baitų skaičius	**1-as baitas**	**2-as baitas**	**3-ias baitas**	**4-as baitas**
0000 0000 0xxx xxxx	1	0xxx xxxx			
0000 0yyy yyxx xxxx	2	110y yyyy	10xx xxxx		
zzzz yyyy yyxx xxxx	3	1110 zzzz	10yy yyyy	10xx xxxx	
1101 10vv vvzz zzyy + 1101 11yy yyxx xxxx	4	1111 0uuu	10uu zzzz	10yy yyyy	10xx xxxx

Pastaba. Lentelėje raidėmis u, v, x, y, z žymimi atskiri kodų bitai ir matyti, kaip jie perkeliami iš 16 bitų į aštuonbičius UTF-8 kodo komponentus. Skaitmenimis 0 ir 1 pažymėti pastovūs bitai, iš kurių galima nustatyti, kiek baitų tam kodui reikės UTF-8 koduotėje.

Vienu baitu koduojami ženklai, kurių unikodo kode pirmieji 9 dvejetainiai ženklai yra nuliai. Tai intervalo 0000–007F arba ASCII ženklai. Jų UTF-8 kodai sutampa su ASCII kodais (xxx xxxx). Todėl UTF-8 kodavimas palankiausias ASCII ženklams.

Dviem baitais koduojami ženklai iš intervalo 0080–07FF. Į jį patenka kiti lotynų rašmenys, taip pat visi graikų, kirilicos, armėnų, hebrajų, arabų, sirų rašmenys, raides modifikuojantys ir diakritiniai ženklai.

Savitųjų lietuvių kalbos abėcėlės raidžių kodavimas UTF-8 būdu pateiktas 19 lentelėje.

Naudojantis aštuonbite žiūrykle, nustatyta „Windows-1257“ koduotei, bus matomos ženklų poros, neturinčios vaizdinio ryšio su koduojama raide. Kai kurie kodai neturi grafinių ženklų aštuonbitėje koduotėje. Jie pažymėti kvadratėliais. Kiti lentelėje parodyti

19 lentelė. Savitųjų lietuvių kalbos abėcėlės raidžių kodavimas UTF-8 būdu

Raidė	**Unikodas**		**UTF-8**		**Raidė**	**Unikodas**		**UTF-8**	
	9...9	**FFFF**	**AA**	**FF FF**		**9...9**	**FFFF**	**AA**	**FF FF**
Ą	260	0104	Ä□	00C4 201E	**ą**	261	0105	Ä...	00C4 2026
Č	268	010C	Ä□	00C4 008C	**č**	269	010D	Ä¨	00C4 00A8
Ę	280	0118	Ä˜	00C4 0098	**ę**	281	0119	Ä™	00C4 2122
Ė	278	0116	Ä–	00C4 2013	**ė**	279	0117	Ä—	00C4 2014
Į	302	012E	Ä®	00C4 00AE	**į**	303	012F	ÄÆ	00C4 00C6
Š	352	0160	Å	00C5 00A0	**š**	353	0161	Å□	00C5 F8FC
Ų	370	0172	Å²	00C5 00B2	**ų**	371	0173	Å³	00C5 00B3
Ū	362	016A	ÅĄ	00C5 0156	**ū**	363	016B	Å«	00C5 00AB
Ž	381	017D	Å½	00C5 00BD	**ž**	382	017E	Å¾	00C5 00BE

grafiniai ženklai gali būti interpretuojami nevienareikšmiškai (pvz., raidės ą kodo antrasis ženklas yra vienas ženklas – daugtaškis, o ne trys taškai). Todėl pateikti ir jų kodai.

Dviem baitais koduojamų ženklų ekonomiškumas toks pat, kaip ir unikodo (du baitai).

Trimis baitais koduojama likusi (ir pati didžiausia) ženklų dalis (iš intervalo 0080–FFFF), išskyrus surogatus (D800–DFFF), kurių poros koduojamos keturiais baitais. Į trijų baitų dalį patenka visų kitų raštų abėcėliniai rašmenys, visi ideografiniai rašmenys, įvairūs matematikos, grafikos ženklai.

Lotynų rašmenis vartojančių kalbų UTF-8 kodavimą lengva atpažinti iš to, kad kiekviena savitoji raidė užkoduota dviem kodais ir vietoj jos tekste matomi du ženklai, neturintys vaizdinio ryšio su užkoduota raide.

ASCII ženklų kodavimas UTF-8 būdu nesiskiria nuo paties ASCII kodavimo 8 bitų koduotėse. Dėl to jis ypatingai palankus šią koduotę vartojančiai anglų kalbai: ir ekonomiškas, ir tekstas normaliai skaitomas bet kuria 8 bitų įranga. Pastaroji savybė gali būti internacionalizavimo ir lokalizavimo klaidų šaltiniu: programos komponentas, veikiantis tik su ASCII koduote, gali būti palaikytas veikiančiu su UTF-8 kodavimu. Todėl į tekstus apdorojančios programos testus būtinai reikia įtraukti raidžių, nesančių ASCII ženklų aibėje.

Kitoms kalboms, vartojančioms lotyniškus rašmenis, šis kodavimas taip pat ekonomiškas, nes dauguma raidžių yra iš ASCII koduotės ir jų kodams pakanka vieno baito. Tekstą galima nesunkiai iššifruoti, bet skirtą normaliam skaitymui naudojantis 8 bitų įranga, reikia iškoduoti. Savitųjų raidžių kodavimas dviem ženklais daro šių raidžių nevisavertiškumo įspūdį.

Kalboms, vartojančioms kitokius rašmenis, kodų ekonomiškumas gali prilygti unikodui (jeigu vienas ženklas koduojamas dviem baitais) arba tekstas pailgėti 1,5 karto, negu unikode (jeigu vienas ženklas koduojamas trim bitais). Bet kuriuo atveju neiškoduotą tekstą skaityti sunku, beveik neįmanoma.

UTF-8 kodavimo būdas yra sklandus atgalinio suderinamumo sprendimas techniniu požiūriu: aiškiai apibrėžtos kodavimo taisyklės, turi aukštą statusą – įtrauktas į unikodą, nesudėtingi kodavimo ir iškodavimo algoritmai, ekonomiškas. UTF-8 daug kur naudojamas, netgi painiojamas su tikru unikodu.

Buvo sugalvotas ir UTF-7 kodavimo būdas kaip 16 bitų koduotės pakaitalas 7 bitų koduote. Buvo tikimasi, kad jis bus naudojamas elektroniniame pašte, bet taip neįvyko.

Apie dalies 32 bitų ženklų kodavimą 16 bitų surogatų poromis rašėme 3.4.1 skyrelyje.

3.5.3. Internacionalizuoti interneto sričių vardai

Interneto sričių vardų sistemos įranga turi du esminius ribojimus: 1) ribota ženklų aibė vardams sudaryti: 26 pagrindinės lotynų abėcėlės raidės (tik mažosios), 10 skaitmenų, brūkšnelis; 2) srities vardo ilgis – nedaugiau kaip 63 ženklai (RFC 1034). Taikant interneto failų vardams priimtą kodavimą kodų sekomis UTF-8 + %FF, vienai ne lotynų raidei reikėtų 6 arba 8 kodų, vienam ideografiniam ženklui – 8 kodų. Taigi varde galėtų būti ne daugiau kaip 10 arba 7 raidės. Tai labai mažai. Todėl buvo ieškoma ekonomiškesnių kodavimo būdų.

Po ilgų svarstymų 2003 metais buvo priimtas RFC 3492 dokumentas, kuriame apibrėžtas specialiai šiam tikslui skirtas kodavimas, angliškai vadinamas „puny“, kas reiškia „mažas“, „silpnas“, „glebus“. Šiuo atveju artimiausia reikšmė, matyt, būtų „mažas“, o gal ir „trumpas“, nes svarbiausias tikslas jį kuriant buvo pakaitos kodo trumpumas, kad jis neviršytų 63 ženklų.

Kiekvienas srities vardas koduojamas atskirai. Ir visas vardas iš karto, ne paraidžiui. Pirmiausiai iš vardo išrenkamos ir į kodą paeiliui surašomos visos varde esančios anglų kalbos abėcėlės raidės, jeigu jų yra. Pavyzdžiui, vardo *abėcėlė* kodo pradžia būtų `abcl`. Toliau analizuojamos ir koduojamos likusios raidės, šiuo atveju *ėėė*. Perkoduojant jos virsta penkiomis raidėmis `wvabb`, kurios prirašomos prie jau esančio kodo pradžios ir atskiriamos brūkšneliu: `abcl-wvabb`.

Tam, kad būtų galima atskirti koduotą vardą nuo tokio pat tikro vardo, parašyto tik anglų kalbos abėcėlės raidėmis, vardo pradžioje prirašoma *xn--*. Ir taip gauname visą perkoduotą vardą:

abėcėlė → `xn--abcl-wvabb`

Koduota vardo dalis neįskaitoma. Bet interneto naršytojas mato tikrus, nekoduotus, vardus. Koduoti vardai matomi serveriuose. Beje, visos interneto tarnybos tikruosius vardus ir jų kodus laiko sinonimais. Todėl jeigu yra problemų su kitų raštų ženklų rinkimu, tai galima pasinaudoti jo kodu. Pavyzdžiui, vietoj *www.*中文*.zh* galima rašyti *www. xn--fiq228c.zh* (internete yra kodavimo ir dekodavimo paslaugas teikiančių svetainių, pvz., „ASP Unicode to Punycode IDN online decoder and encoder“[1]).

Trumpai apie šį kodavimą. Skaičiuojami skirtumai tarp ženklų kodų unikode ir tie skirtumai išreiškiami skaičiais, koduotais 36 ženklais (a...z, 0...9). Todėl daugiau-

[1] ASP Unicode to Punycode IDN online decoder and encoder, http://www.motobit.com/util/punycode-decoder-encoder.asp

sia ženklų kode prireikia pirmajai raidei. Kitos to paties rašto raidės paprastai būna netoliese, atstumai mažesni, skaičiai trumpesni, jiems reikia mažiau ženklų. Pavyzdžiui, žodžio *abėcėlė* kodas būtų *abcl-wvab*, t. y. paskutinei žodžio *abėcėlė* raidei *ė* prireikė tik vienos papildomos raidės *b*. Kaip kodas darosi vis ekonomiškesnis didėjant raidžių skaičiui varde parodysime pavyzdžiais:

ą → `xn--2da` (1 raidė → 7 ženklai)
ąčęėįšųūž → `xn--2daq4ah5ionmc1c6e` (9 raidės → 21 ženklas)
ąčęėįšųūžąčęėįšųūžąčęėįšųūžąčęėįšųūžąčęėįšų
→`xn--`
`2daaaaa8bbbbb6hcccctdddd16beeee49jfafff00aggg2xhhhh32biaii`
(43 raidės → 62 ženklai, dar neviršijantys 63 ženklų ribos).

Eksperimentuojant su kinų raštu pavyko užkoduoti, neviršijant 63 ženklų, po 20–30 ideografinių ženklų – kadangi jų daug, tai ir atstumai tarp jų kodų didesni, todėl greičiau pasiekiama 63 ženklų riba.

Šis kodavimas internete įsigaliojo ir internacionalizuoti vardai buvo pradėti registruoti nuo 2003 m. kovo mėnesio. Nuo 2004 m. kovo 30 d. srityje *lt* pradėti registruoti vardai, turintys savitųjų lietuvių kalbos abėcėlės raidžių. 2010 metais veikė apie tūkstantis tokių vardų, t. y. apie 0,8 proc. visų šioje srityje užregistruotų vardų turėjo savitųjų raidžių. Lietuviški vardai taip pat registruojami *biz, com, eu, info, net, org* ir kitose srityse.

Nacionalinių aukščiausiojo lygio sričių vardai sutampa su valstybių kodais. Valstybėms, kurių kalbos vartoja ne lotyniškus rašmenis, derėtų turėti aukščiausiojo lygio sritį, užrašytą savais rašmenimis. Interneto vardų ir numerių skirstymo korporacija ICANN nuo 2009 m. lapkričio 16 d. pradėjo priimti paraiškas aukščiausiojo lygio sričių internacionalizuotiems vardams. Tokių vardų 2010 metais jau buvo užregistruota kelios dešimtys – juos jau turi Rusija, Kinija, Tailandas, Saudo Arabija ir kitos valstybės.

Internacionalizuoti vardai pradėti vartoti neseniai. Interneto naršyklės šiais vardais pavadintus tinklalapius sėkmingai atveria. Tačiau mažiau matomose vietose, juo labiau programose, kurios nėra skirtos naršyti, bet gali nusiųsti saitą interneto naršyklei, kad ji atvertų tinklalapį (pvz., tekstų rengyklėse, žiūryklėse), dar gali būti nesklandumų. Pasitaiko atvejų, kai vietoj tikro vardo rodomas koduotas arba naršyklei nusiunčiamas neteisingas adresas ir ši negali atverti tinklalapio.

3.6. Ženklų įvedimas ir vaizdavimas

Kalbėdami apie kompiuterio pritaikymą konkrečios kultūros ir kalbos terpei paprastai turime omenyje programinę įrangą. Tačiau yra vienas svarbus aparatinės įrangos komponentas, kuris tai terpei taip pat turi tikti. Tai – klaviatūra. Sakome „tikti“, o ne „pritaikyti“, nes šiuolaikinių kompiuterių klaviatūrų užrašai ant klavišų yra graviruojami ir tą darbą atlieka klaviatūros gamintojas, pavaizduodamas užrašus ant švarių klavišų. Daugelio kalbų klaviatūros tvarkyklės jau būna įdėtos į operacinę sistemą, net ir nelokalizuotą.

Graviravimą galima laikyti ir lokalizavimu, kurį atlieka klaviatūros gamintojas. Jis ima kalbos požiūriu neutralų klaviatūros korpusą ir švarius klavišus, ant klavišų uždeda užrašus, klavišus sumontuoja į jiems tinkamas vietas, uždeda tinkamus užrašus ant klaviatūros korpuso.

Užrašas ant klavišo – tai informacija žmogui, vienareikšmiškai nustatanti, ką jis gaus paspaudęs tą klavišą. Su kiekvienu klavišo paspaudimu ir atleidimu į kompiuterį siunčiamas klavišo kodas, identifikuojantis klavišo vietą (koordinatę) klaviatūroje ir su tuo klavišu atliktą veiksmą (paspaudimą arba atleidimą). Kokį ženklą ar komandą atitinka tas klavišas, nustato klaviatūros tvarkyklė. Taip susiejami užrašai ant klavišų su jų siunčiamais ženklais į kompiuterį.

Klaviatūros tvarkyklę sudaro dvi dalys: vykdomoji ir informacinė. Vykdomoji dalis yra viena bendra visoms klaviatūroms (kalboms). Informacinėje dalyje pateikiama informacija apie tai, kurie ženklai išdėstyti ant kurių klavišų. Ši dalis atskira kiekvienai klaviatūrai.

Informacinės dalys būna sudėtos į operacinę sistemą. Kai pasirenkama klaviatūra faktiškai pasirenkama jos informacinė dalis. Todėl dažnai ji vadinama tiesiog klaviatūros tvarkykle.

Informacinės dalies atskyrime nuo vykdomosios galima įžiūrėti analogiją su lokalizuojamųjų išteklių atskyrimu nuo vykdomosios programos dalies. Bet skiriasi tuo, kad informacinė dalis sukuriama kiekvienai klaviatūrai iš naujo, dažniausiai automatizuotai, o ne gaunama lokalizuojant kitos klaviatūros tvarkyklę.

3.6.1. Ženklų aibė ir jos išdėstymas klaviatūroje

ASCII koduotėje yra 94 rašto ženklai (neskaitant tarpo). Rašomųjų mašinėlių klaviatūros turėjo du lygius (registrus), t. y. kiekvienam klavišui priskiriami du ženklai.

Amerikiečiai, savo rašomosios mašinėlės klaviatūros klavišų skaičių padidinę iki 47, gavo JAV QWERTY klaviatūrą, visiškai atitinkančią ASCII koduotę.

Tokia klaviatūra netiko kitoms valstybėms, netgi anglų kalbą vartojančiai Jungtinei Karalystei – reikėjo dar vieno ženklo svarui sterlingų, o išmesti dolerio ženklo nebuvo kaip. Čia reikia prisiminti koduočių istoriją, kur valiutos ženklas buvo labai svarbus ginčų objektas. Taip atsirado 48 klavišų klaviatūra. Jos fizinis dydis toks pat, kaip ir 47 klavišų klaviatūros, o 48-as klavišas gautas sutrumpinus kairįjį antrojo lygio klavišą.

Klavišai skaičiuojami dar ir kitaip – įskaitant ir valdymo klavišus. Tada 48 klavišų klaviatūra vadinama 102 klavišų klaviatūra, o 47 klavišų – 101 klavišo klaviatūra.

Kitų kalbų rašto ženklams nepakako ir 48 klavišų. Lotyniškus rašmenis vartojančioms kalboms kiekvienai savitajai abėcėlės raidei reikia atskiro klavišo, o atsisakyti kurio nors toje kalboje nevartojamo ženklo (pvz., @, #, &), bet esančio ASCII koduotėje, rizikinga – jeigu ženklas ten yra, tai jis bus panaudotas ir kurioje nors programoje. Todėl buvo pasinaudota kita, efektyvesne, ženklų skaičiaus didinimo galimybe – klaviatūros papildymu trečiuoju lygiu, t. y. klavišui priskiriant dar vieną ženklą. Šiam lygiui priklausantis ženklas gaunamas paspaudus trečiojo lygio klavišą *Lyg3* (daugelyje kitų kalbų klaviatūrų yra istoriškai išlikęs užrašas *Alt* arba *AltGr* – nuo tų laikų, kai trečiojo lygio dar nebuvo, o klavišas jau buvo ir buvo naudojamas kaip alternatyvos klavišo dublikatas dešinėje klaviatūros pusėje).

Trečiojo lygio ženklai piešiami ant klavišo dešinėje, žemai.

Ženklų rinkimo patogumas ir sparta priklauso nuo lygio. Pirmojo lygio ženklai gaunami paspaudus tik vieną to ženklo klavišą. Antrojo – du klavišus: to ženklo ir (laikant paspaustą) antrojo lygio klavišą *Lyg2*, kairįjį arba dešinįjį. Šiokią tokią privilegiją turi didžiosios raidės. Ištisinį jų tekstą galima rinkti vieno klavišo paspaudimais esant didžiųjų raidžių būsenoje (prieš tai paspaudus *Didž* klavišą). Trečiojo lygio ženklai gaunami taip pat dviem klavišais: to ženklo ir laikant paspaustą trečiojo lygio klavišą *Lyg3*, kuris yra tik vienoje klaviatūros pusėje (dešinėje) ir trumpesnis už antrojo lygio klavišus. Dėl to surinkti šiuos ženklus mažiausiai patogu.

Pirmieji du kompiuterio klaviatūros lygiai paprastai paliekami tokie pat, kaip ir rašomojoje mašinėlėje arba nežymiai pakoreguojamas kai kurių specialiųjų ženklų išdėstymas. Į trečiąjį sudedami rečiau vartojami ženklai.

Kada kuri valstybė pradėjo naudoti savą kompiuterio klaviatūrą dabar sunku atsekti. Tačiau 1991 metais išleistame MS DOS žinyne (DTK, 1991) iš pateiktų 20 klaviatūrų 19 (išskyrus JAV) turi 48 klavišus ir 18 (išskyrus JAV ir Jungtinės Karalystės) turi trečiąjį lygį. Sąraše buvo tuometinio Rytų bloko valstybių Čekoslovakijos (čekų

ir slovakų), Jugoslavijos, Lenkijos ir Vengrijos klaviatūros. Baltijos valstybių šiame žinyne dar nebuvo. Mūsų žiniomis 1994 metais estai jau turėjo trijų lygių klaviatūrą ir ja komplektavo kompiuterius.

Reikalavimas, kad klaviatūra turėtų tris lygius ir 48 ženklų klavišus yra įformintas tarptautiniuose standartuose ISO 9995-1–7. Juose išdėstyti ir kiti bendrieji reikalavimai, kurių turėtų laikytis nacionaliniai standartai. Į lotyniškus rašmenis vartojančių kalbų klaviatūras turi būti įtraukti visi ISO/IEC 646 standarto invariantinės dalies ženklai, skaitmenys turi būti išdėstyti viršutinėje klavišų eilėje. Nelotyniškus rašmenis vartojančioms kalboms įvedama antra ženklų grupė (papildoma grupė lotyniškiems rašmenims). Grupė – tai klaviatūros būsena. Jai perjungti klaviatūroje turi būti atskiras klavišas, o indikacijai – signalinė lemputė. Ženklų išdėstymo, tekstinių užrašų ant klavišų ir daugelio kitų dalykų šis standartas nereglamentuoja – tai palikta nacionalinių standartų kompetencijai.

Lietuvoje kartu su pirmaisiais asmeniniais kompiuteriais pradėta naudoti JAV klaviatūra ir tvarkyklė, kurioje devyni viršutinės eilės klavišai (skaitmenys 1–8 ir dešimt kitų ženklų) pakeičiami savitosiomis lietuvių kalbos abėcėlės raidėmis (dėl to ji vadinama *skaičiukine*). Tai buvo paprastas būdas namudinėmis sąlygomis klaviatūrą pritaikyti lietuviškiems rašmenims įvesti. Bet taip pakoreguota klaviatūra buvo nevisavertė, nes nebuvo galima surinkti minėtomis raidėmis „uždengtų“ ženklų.

1992 metais buvo priimtas Lietuvos standartas LST 1205-92, kuriame savitosios lietuvių kalbos abėcėlės raidės išdėstytos taip, kaip ir rašomojoje mašinėlėje, išskyrus tik tai, kad raidžių F ir Š klavišai sukeisti vietomis. Tačiau klaviatūra buvo tik dviejų lygių ir turėjo 47 klavišus. Teko atsisakyti aštuoniolikos ASCII ženklų. Dėl to ji taip pat nebuvo visavertė.

2000 metais buvo priimtas lietuviškos klaviatūros standartas LST 1582, atitinkantis tarptautinį standartą ISO/IEC 9995 ir apibrėžiantis klaviatūrą, analogišką kitų Europos kalbų, vartojančių lotyniškus rašmenis, klaviatūroms (išskyrus latvių). Tai 48 ženklų, trijų lygių klaviatūra (4 priedas, 59 pav.). Palyginimui pateiktos vokiečių ir prancūzų kompiuterių klaviatūros (4 priedas, 60 ir 61 pav.).

Standarte pasirinktas tradicinis lietuviškas ženklų išdėstymas ĄŽERTY. Pagrindinės lotynų abėcėlės raidžių išdėstymas nedaug kuo skiriasi nuo QWERTY – tik keturiomis raidėmis F, Q, W, X. Toks mažas skirtumas negali pastebimai padidinti jos ergonomiškumo. Standarto tikslas ir nebuvo siekti absoliutaus ergonomiškumo, nes tokiu atveju reikėtų nutolti nuo QWERTY, tiksliau – išvis jo nesilaikyti, o tai sukeltų nepatogumų naudojantis kitų kalbų klaviatūromis. Tikslas buvo parengti visavertę klaviatūrą, kurioje būtų galima rinkti lietuvišką tekstą ir visus kitus kompiuterio

naudotojui reikalingus ženklus, savitąsias lietuvių kalbos abėcėlės raides laikant lygiavertėmis kitoms klaviatūroje esančioms raidėms (Tumasonis, Grigas, 2000).

Rengiant standartinę klaviatūrą buvo numatyta galimybė kirčiuotas raides įvesti naudojantis ta pačia fizine klaviatūra. Šiam tikslui buvo numatytas vienas klavišas (viršuje kairėje) su visais trimis kirčio ženklais. Vėliau buvo parengta ir kirčiuotų raidžių klaviatūros tvarkyklė (Tumasonis, 2007).

Į operacines sistemas būna įdėta po kelias klaviatūros tvarkykles vienai kalbai, dažniausiai skirtas įvairiems specifiniams poreikiams (pvz., ergonomiškos, fonetinės) arba dėl atgalinio suderinamumo su ankstesne įranga. Operacinės sistemos lokalizuotojas turėtų nustatyti tai kalbai tinkamiausią klaviatūrą numatytąja. Kitokią gali pasirinkti specifinius poreikius turintis kompiuterio naudotojas.

Klavišų pavadinimai. Trumpi klavišų pavadinimai ir (arba) piktogramos užrašomi ant klavišų. Tekstuose, matomuose ekrane, o taip pat popierinėje dokumentacijoje tokio pavadinimo ne visada užtenka. Kartais reikia pasakyti, kurį klavišą iš alternatyvių reikia paspausti, pavyzdžiui, kairįjį ar dešinįjį valdymo klavišą (jų funkcijos gali būti skirtingos), skaitmens klavišą, esantį viršutinėje pagrindinės klaviatūros dalyje ar skaitmenų dalyje dešinėje. Kartais reikalingi trumpi, formulių pavidalo užrašai, pavyzdžiui, *Vald + 5 (sk.)*, kas reiškia, kad skaitmens 5 klavišą reikia imti iš skaitmeninės klaviatūros dalies.

Išsamūs ir trumpi klavišų pavadinimai pateikti 2 priede.

„Windows“ operacinėse sistemose, pradedant „Windows XP“ laida, trumpi klavišų pavadinimai įtraukiami į klaviatūrų tvarkyklių informacinę dalį. Tokioje sistemoje veikianti programa gali pati pasiimti iš klaviatūros tvarkyklės reikiamo klavišo pavadinimą ir įtraukti į ekrane rodomą tekstą. Tada nebereikia jo dėti į lokalizuojamuosius išteklius, o lokalizuotojui – versti. Atkrenta pavadinimų vienodinimo problema ir nebelieka su tuo susijusių klaidų. Tačiau pavadinimų įdėjimu į tvarkyklę turi pasirūpinti tvarkyklės autorius. Taip šis uždavinys perkeliamas į internacionalizacijos lygmenį. Lokalizuotojui lieka išaiškinti galimas internacionalizacijos klaidas.

3.6.2. Komandų klavišai

Daugelis komandų (sparčiųjų) klavišų derinių gaunama paspaudus kurį nors valdymo klavišą, dažniausiai *Vald*, kartu su kuriuo nors ženklo klavišu. Kadangi ženklų klavišų išdėstymas skirtingų kalbų klaviatūrose skiriasi, tai pasitaiko internacionalizacijos klaidų.

Klavišų identifikavimas, skirtas deriniams gauti, tvarkyklėse pateikiamas atskirai nuo ženklų priskyrimo klavišams. Todėl tvarkyklės autoriui lengva suklysti – klavišo identifikavimui netyčia priskyrus kitą ženklą, negu juo išgaunamą. Juolab, kad tas identifikavimas gana painus. Pasitaiko, kad tvarkyklės autorius visą identifikuojamųjų ženklų eilutę nukopijuoja iš kitos klaviatūros tvarkyklės su kitokiu ženklų išdėstymu. Taip tvarkyklėje atsiranda klaidų.

Raidžių klavišai identifikuojami didžiosiomis raidėmis, nors spaudžiame taip, lyg rinkdami mažąją (be antrojo lygio klavišo, esant neįjungtai didžiųjų raidžių būsenai). *Vald+Lyg2+raidė* – tai nauja komandų klavišų kombinacija, kurioje *Lyg2* klavišas figūruoja kaip savarankiškas komandos klavišų derinio komponentas, o ne raidės lygio keitiklis (dėl to pvz., *Vald+C* kopijavimo veiksmą atliks, o *Vald+Lyg2+C* jo neatliks).

Skaitmenų klavišai identifikuojami skaitmenimis, nesvarbu, koks kitas ženklas yra ant to klavišo.

Problema atsiranda, kai ant to paties klavišo abu (pirmojo ir antrojo lygio) ženklai yra specialieji (ne raidės ir ne skaitmenys). Paprastai klavišas identifikuojamas vieno kurio nors lygio ženklu, o kito lygio ženklas lieka nepanaudotas. Bet skirtingų kalbų klaviatūrose specialieji ženklai gali būti suporuoti skirtingai. Jeigu toks skirtumas yra tarp originalios programos ir lokalizacijos klaviatūrų, tai lokalizuota programa „nežinos“, kurį ženklą atitinkančią komandą ji turi vykdyti. Jeigu komandos klavišų formavimas yra lokalizuojamuose ištekliuose, tai situaciją gali (ir turi) ištaisyti lokalizuotojas. Priešingu atveju tai reikia laikyti internacionalizacijos klaida.

Dar viena kartais pasitaikanti internacionalizacijos klaida yra trečiojo lygio klavišo panaudojimas komandų klavišų deriniams gauti. Tais laikais, kai trijų lygių klaviatūrų dar nebuvo, dešinysis *Alt* buvo naudojamas komandos klavišų deriniui gauti. Kai toks derinys yra programoje, tai jį surinkus vykdoma toje programoje nustatyta komanda, o trečiojo lygio ženklas užblokuojamas, t. y. toje programoje jo negalima surinkti.

Tokia pat klaida būna, kai originalioje programoje panaudojamas klavišų derinys *Ctrl+Alt+ženklas*. Mat anksčiau, kai klaviatūrose dar nebuvo dešiniojo *Alt* klavišo, tai trečiojo lygio klavišas buvo modeliuojamas klavišų pora *Ctrl+Alt*. Tokia galimybė yra ir dabar (atgalinis suderinamumas!). Todėl šių dviejų klavišų derinio panaudojimas yra ta pati internacionalizacijos klaida, kaip ir su vienu dešiniuoju *Alt*.

Šiame skyrelyje vartojome angliškus klavišų pavadinimus dėl to, kad lietuviškas klavišo žymuo *Lyg3* aiškiai nusako klavišo paskirtį ir ji intuityviai suvokiama. Lietuviui programuotojui būtų sunku suvokti, kaip tokią klaidą išvis galima padaryti.

3.6.3. Klaviatūros įtaka rašto kultūrai

Dauguma tekstų į kompiuterių pasaulį kol kas patenka per klaviatūrą (išskyrus retesnius atvejus, kai naudojami spausdinto teksto arba balso automatinio atpažinimo įrenginiai). Todėl teksto ir atskirų rašto ženklų rinkimo paprastumas ir patogumas gali turėti įtakos rašto kultūrai.

Skaitydami kompiuterinius tekstus neretai pastebime, jog mažąja raide pradedami miestų, valstybių pavadinimai, netgi savo asmenvardžiai, vietoj raidžių su diakritiniu ženklu vartojamos į jas panašios be diakritinio ženklo. Daugelis teigia, kad taip elgiamasi dėl to, kad savitąsias kalbos raides sunkiau surinkti. Norint patikrinti, ar taip iš tikrųjų yra, buvo atliktas eksperimentas – tokių klaidų analizė pokalbių programos „Skype“ abonentų registracijos duomenyse (Grigas, Pedzevičienė, 2009).

Analizės rezultatai parodė, kad savo miestų vardus iš mažųjų raidžių rašo nuo 5 iki 22 proc. įvairių tautų atstovų (5 proc. islandų ir vokiečių, 6 proc. čekų, ir t. t.). Mat didžiajai raidei surinkti bet kurios valstybės klaviatūra reikia šiek tiek daugiau pastangų: paspausti dar ir antrojo lygio klavišą. Todėl kai kam kyla pagunda vietoj jos rinkti mažąją.

Raidžių su diakritiniais ženklais ir kitų savitųjų raidžių rinkimo sudėtingumas skirtingų kalbų klaviatūrose skirtingas. Todėl naudojamos klaviatūros buvo suskirstytos į tris grupes: 1) originalios, kuriose visos savitosios raidės išdėstytos vienodose pozicijose su kitomis raidėmis (danų, estų, suomių, švedų, vokiečių); 2) originalios, bet dalies savitųjų raidžių rinkimas šiek tiek sudėtingesnis (čekų, islandų, ispanų, lenkų, prancūzų); 3) pritaikytos kitos kalbos fizinės klaviatūros, kuriose visų savitųjų raidžių rinkimas sudėtingesnis (latvių, lietuvių skaičiukinė).

Pirmosios grupės klaviatūrų naudotojai darė 0–8 proc. klaidų, antrosios – 13–55 proc., trečiosios – 77–83 proc.

Prie trečiosios grupės klaviatūrų rezultatų bloginimo prisideda dar ir tas faktas, kad naudojant kitos kalbos klaviatūrą ant klavišų paliekami originalios kalbos užrašai, nebeatitinkantys lokalizacijos kalbos tvarkyklės, o lokalizacijos kalbos raidės užrašomos kitose vietose, negu turėtų būti pagal tos kalbos tvarkyklę, o kartais ir visai neužrašomos. Tai formuoja įvaizdį, kad savitosios lokalizacijos kalbos raidės nevisavertės, o gal ir kalba nevisavertė. Tikrovės neatitinkantys užrašai yra rimta, kartais netgi sunkiai įveikiama kliūtis pradedančiajam arba atsitiktiniam kompiuterio naudotojui.

Klaviatūra, kaip ir lokalizuota programinė įranga, turi būti tokia, lyg ji būtų specialiai sukurta tai kalbos ir kultūros aplinkai. Norint tai pasiekti programinė įranga lokalizuojama, nes lokalizuoti daug kartų pigiau, negu pagaminti originalią. Klaviatūros gamybos kaina nepriklauso nuo to, kokie ženklai užrašyti ant klavišų. Todėl klaviatūros iš karto gaminamos konkrečiai kalbai. Programinės įrangos lokalizuotojui svarbu pasirūpinti, kad lokalizacijos programinė įranga būtų pritaikyta darbui su konkrečiai kalbai tinkama klaviatūra.

Į operacines sistemas būna įdėta daugelio kalbų klaviatūrų tvarkyklės. Prireikus rinkti kitos kalbos tekstą, galima persijungti į tos kalbos klaviatūrą. Nelabai patogu, nes daugelis užrašų ant klavišų nebeatitinka jais išgaunamų ženklų, bet surinkti galima visus ženklus. Jeigu kompiuteris sukomplektuotas su 47 klavišų klaviatūra, tai tada tų ženklų, kurie gaunami su 48-uoju klavišu surinkti negalima (nėra ką paspausti...). Taigi kompiuterio komplektavimas 47 klavišų klaviatūra, kas būdinga skaičiukinės klaviatūros atveju, kitų Europos klaviatūrų naudojimą paverčia fikcija.

3.6.4. Kodavimas mobiliuosiuose telefonuose

Trumpųjų žinučių kodavimo telefonuose istorija neilga, bet pakankamai prieštaringa. Pirmoji žinutė iš kompiuterio į mobilųjį telefoną buvo išsiųsta 1992 m. gruodžio 3 d. Jungtinėje Karalystėje, o iš mobiliojo telefono į mobilųjį – 1993 metais Suomijoje. Tada kompiuteriuose jau buvo vartojamos aštuonių bitų koduotės. Tačiau numatytoje telefonų koduotėje *GSM 03.38* buvo apsiribota 7 bitais, šiek tiek padidinus rašto ženklų skaičių lyginant su ASCII koduote, nes telefonuose nereikėjo valdymo kodų ir jie buvo atiduoti ženklams. Sutaupyti kodai buvo panaudoti papildomoms raidėms. Į koduotę buvo įtraukta 9 didžiosios ir 14 mažųjų lotynų abėcėlės raidžių, 10 didžiųjų graikų abėcėlės raidžių (20 lentelė).

20 lentelė. Papildomos raidės GSM 03.38 koduotėje

Ženklų grupė	**Ženklų skaičius**	**Ženklai**
Papildomos lotynų abėcėlės raidės	23	Ä ä Å å à Æ æ Ç É é è ì Ñ ñ ò Ø ø Ö ö ß Ü ü ù
Graikų kalbos abėcėlės raidės	10	Δ Φ Γ Λ Ω Π Ψ Σ Θ Ξ

Kodų vis tik buvo per mažai. Keletas lotynų abėcėlės mažųjų raidžių liko be jas atitinkančių didžiųjų, viena didžioji – be mažosios. Apsiribota tik dešimčia didžiųjų graikų kalbos abėcėlės raidžių. Atseit, vietoj kitų didžiųjų, kurių piešiniai sutampa su lotynų abėcėlės raidėmis, galima vartoti lotyniškas. Visai taip, kaip pirmose rusiškose koduotėse, kuriose nebuvo skiriamos lotyniškos raidės nuo tokius pat piešinius turinčių kirilicos raidžių.

Tuo metu egzistavę standartai ir rekomendacijos (ETSI ETS 300 640, 1996; ITU-T Recommendation E.161, 2001 (pirmasis leidimas 1995)) numatė tik 26 anglų kalbos abėcėlės raidžių išdėstymą ant skaitmenų klavišų.

Taigi tie nedideli raidžių papildymai problemos neišsprendė, tik atitolino jos sprendimą. Dėl to septynių bitų koduotė telefonuose išsilaikė ilgą laiką, darydama neigiamą įtaką kalbų, kurių abėcėlių raidės nepateko į tą ribotą septynių bitų koduotę, kultūrai.

Tik 2003 metais buvo priimtas ETSI ES 202 130 standartas, reglamentuojantis žinučių kodavimą unikodu. Pirmame standarto leidime (ETSI ES 202 130, 2003) apibrėžti mobiliuosiuose telefonuose vartotini 28 Europos kalbų rašto ženklų rinkiniai. Tarp jų – ir lietuvių.

Kiekvienos kalbos raidės skirstomos į dvi grupes: A ir B. A grupės raidės yra privalomos kalbos abėcėlės raidės, B grupės – antraeilės, kurių gali prireikti, pavyzdžiui, užsieniniams asmenvardžiams užrašyti. A grupėje yra visos 32 lietuvių kalbos abėcėlės raidės, B grupėje – Q, W, X.

Standarte apibrėžiamas ženklų rikiavimas kiekvienai kalbai atskirai ir bendras visoms kalboms daugiakalbiame tekste.

Nustatytas ženklų priskyrimas skaitmenų klavišams.

Visa tai sudaro prielaidas tvarkingai rašyti žinučių tekstus ir operuoti su jais analogiškai, kaip ir su kompiuteriniais tekstais.

Perėjimas nuo septynių bitų tiesiai prie unikodo buvo didelis šuolis į priekį, taip aplenkta daugelis programų, kol kas apsiribojančių aštuonių bitų koduotėmis ar aštuonbičiu unikodo pakaitalu UTF-8.

Standartas buvo toliau tobulinamas, įtraukiamos naujos Europoje vartojamos kalbos, ne tik joje gyvenančių tautų, bet ir emigrantų (Böcker, Larsson, 2006). 2007 metais buvo išleistas antrasis jo leidimas (ETSI ES 202 130, 2003). Pagrindiniai principai išliko tie patys, kalbų skaičius padidintas iki 79.

Priimant standartą buvo diskusijų dėl raidžių loginio išdėstymo ant klavišų: ar visas vienam klavišui priskirtas abėcėlės raides išdėstyti prieš skaitmenį, ar dalį jų –

konkrečiai – raides, nesančias anglų kalbos abėcėlėje, nukelti po skaitmens (Böcker, Larsson, 2006; Böcker, Larsson, 2007). Buvo nutarta lotynų rašmenis vartojančių kalbų raides, nesančias anglų kalbos abėcėlėje, nukelti po skaitmens, nors logiškiau būtų išdėstyti visas prieš skaitmenį, pagal abėcėlę. Remtasi abejotinu teiginiu, kad visų raidžių išdėstymas prieš skaitmenį mažiau patogus silpnaregiams (Böcker, Larsson, 2006).

Prieš skaitmenį raidės išdėstomos „Nokia“ telefonuose. Rekomendacijose telefonų lokalizuotojams į lietuvių kalbą[1] taip pat teikiamas šis išdėstymas.

Dalies raidžių nukėlimas po skaitmens prieštarauja ir paties ETSI ES 202 130 standarto dvasiai: neatitinka raidžių skirstymo į A ir B grupes, atsiranda skirtumas tarp lotyniškus rašmenis vartojančių kalbų ir kitokius rašmenis vartojančių.

Šio trūkumo galima išvengti pasirenkant telefoną. Problemų su žinučių persiuntimu tinklais išvis nėra, nes visame pasaulyje naudojama viena ir ta pati koduotė – unikodas. Lokalizuotojui tereikia pasirūpinti, kad būtų galima surinkti visas lietuvių kalbos raides, o jeigu įdėta galimybė pasirinkti kelias žinučių rašymo kalbas, tai lietuvių kalbą nustatyti numatytąja.

Dar likęs problematiškas dalykas – užrašai ant klavišų. Lotyniškus rašmenis vartojančių kalbų telefonuose vadovaujamasi Tarptautinės telekomunikacijų sąjungos ITU rekomendacijomis (ITU-T Recommendation E.161, 2001), kuriose nustatytas tik anglų kalbos abėcėlės raidžių išdėstymas. Tokie užrašai kitų kalbų telefonų savininkams gali sukelti nepatogumų (Böcker, Larsson, 2006), nes pažeidžiamas principas „ką matau, tą gaunu“.

Minėtos rekomendacijos nėra blogos. Tik jos taikomos ne tam, kam buvo skirtos. Jos skirtos telefono numerio įsiminimui ir rinkimui pagerinti, kad žmogus galėtų vietoj telefono skaitmeninio numerio įsiminti ir rinkti raidinį kodą. Pavyzdžiui, Jonas, pasistengęs gauti telefono numerį tokį, kad jo skaitmenys atitiktų vardo raides, t. y. 56627, galėtų vietoj numerio sakyti: skambinkite man telefonu Jonas. Aišku, kad čia reikalingas vieningas tarptautinis raidžių priskyrimas skaitmenims, kad bet kuriame telefone raidė J būtų ant skaitmens 5, raidė O – ant skaitmens 6 ir t. t.

Renkant žinutes reikia gauti ne skaitmenis, o raides. Todėl pirminis elementas yra raidė ir čia minėtų rekomendacijų taikymas neatitinka jų paskirties.

[1] Specifics of Lithuanian language loacalization for mobile phones (2007). Guidelines for implementers. Lithuanian Mobile Phone User Group. http://unicode.strangled.net/mobile/Specifics of Lithuanian language localization for mobile phones.pdf

4. LOKALĖ

Kiekviena vietovė (pvz., valstybė ar jos dalis) turi nusistovėjusių kultūrinių normų visumą. Šių normų laikomasi kasdieniniame gyvenime: kalbant, rašant, bendraujant. Kadangi informacinės technologijos yra neatsiejama šiuolaikinio gyvenimo dalis, tai suprantama, kad kompiuteriu turėtų būti galima rengti kultūrines normas atitinkančius dokumentus ir dialogas su kompiuteriu konkrečioje kultūrinėje terpėje vyktų laikantis tokių pačių normų – tai nusako lokalė.

4.1. Lokalės samprata

Lokalė – tai programinės įrangos lokalizavimo pagrindinė koncepcija. Tarptautiniame ir Lietuvos kultūrinių nuostatų registravimo procedūrų standarte LST ISO/IEC 15897 lokalė apibrėžiama kaip „naudotojo aplinkos poaibio, priklausančio nuo kalbos ar kultūrinių normų, apibrėžimas“. Tačiau ne visas kultūrines normas galima vienareikšmiškai apibrėžti, o tokių normų svarba yra neabejotina kuriant ar lokalizuojant programinę įrangą. Todėl dažnai lokalė yra suprantama plačiau, kaip visų nuo konkrečios vietovės (ar tai būtų valstybė, ar jos dalis) ir kalbos priklausančių dialogo su naudotoju aspektų visuma, dažnai pabrėžiamas kultūrinių normų vartojimas kompiuteryje ir programinėje įrangoje, pavyzdžiui, lokalę galime apibrėžti kaip kompiuteryje ir jo programinėje įrangoje vartojamų elementų, priklausančių nuo kalbos ir kultūros normų, visumą.

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 formaliai apra-

šant lokales paprastai apsiribojama apibendrinta kultūrinių aspektų aibe (akivaizdžiais kultūriniais faktoriais). Pagrindinės dalys, kurias apima dauguma lokalių, yra:

- **Kalba ir šalis**. Kokia kalba naudotojas bendrauja 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, datos ir laiko, valiutos, adresų, telefono numerių). Kaip 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 programinės įrangos gamintojai naudojo skirtingus metodus kultūrinėms nuostatoms aprašyti, todėl atsirado keletas lokalių modelių, kurie nėra tarpusavyje iki galo 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 (Laucius, 2003).

Pagrindiniai lokalę identifikuojantys elementai yra kalba ir valstybė (teritorija). 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.

Dažniausiai naudojami dviraidžiai kalbų kodai, apibrėžti standarte ISO 639-1, ir dviraidžiai valstybių arba vietovių kodai, apibrėžti standarte ISO 3166-1. Tokią lokalių identifikaciją nusako RFC 3066 ir XPG4 dokumentai, ir ji yra plačiai taikoma. Lietuvos lokalė paprastai nurodoma eilute „lt-LT“, JAV anglų – „en-US“, Jungtinės Karalystės anglų – „en-GB“ ir t. t. Tačiau „Windows“ aplinkoje lokalės identifikuojamos skaitiniais lokalių identifikatoriais LCID[1] (pagal XPG4 lietuvių lokalė identifikuojama eilute „lt-LT“, o pagal LCID – skaičiumi 42716) (Tuoc ir kt., 1995).

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

Lokales programinėje įrangoje paprasčiausia realizuoti pasinaudojant operacinės sistemos ar kitos platformos lokalių paslaugomis. Tačiau šis būdas nėra universalus dėl dviejų svarbiausių priežasčių:

1. Lokalėje apibrėžiama kalbos abėcėlė, ženklų koduotės, ženklų rikiavimas, datos ir laiko žymėjimas, ženklų išdėstymas klaviatūroje, pašto adresų forma, laiškų forma, kreipiniai į asmenis ir daug kitų konkrečiai kalbai ir kultūrai būdingų dalykų. Kiekviena kalba ir valstybė gali turėti savo lokalę. Viena valstybė, kurioje yra vartojamos įvairios kalbos, gali turėti kelias lokales (pvz., Šveicarijos vokiečių ir Šveicarijos prancūzų). Ta pati kalba, vartojama keliose valstybėse, taip pat gali turėti atskiras lokales (pvz., Prancūzijos prancūzų ir Kanados prancūzų).
2. Lokalių elementai, su kuriais dažniausiai susiduriama programinėje įrangoje, yra kaupiami, ieškoma formalių būdų jiems pateikti, standartizuoti jų pateikimą ir naudojimą (Laucius, 2003). Be abejo, visų lokalės elementų neįmanoma aprašyti formaliai, pavyzdžiui, kultūrai būdingo mąstymo stiliaus įtakojamų programinės įrangos funkcijų, grafikos elementų, spalvų vartojimo, tam tikros kalbos eilučių visų komponavimo atvejų, visų žodžių formų, todėl formalizuojami tik pagrindiniai lokalių elementai.

Vieną programinės įrangos lokalizavimo dalį sudaro dialogo tekstų (programos languose ir jų elementuose – ant mygtukų, etikečių, meniu ir pan. – matomų užrašų, pranešimų) bei elektroninių žinynų ir kitos, su programa susijusios, medžiagos vertimas į lokalės kalbą. Kita ne mažiau svarbi programinės įrangos lokalizavimo dalis – tai kitų programoje naudojamų su lokale susijusių elementų adaptavimas.

4.2. Pagrindiniai kultūros elementai

Skirtingi kultūros apibrėžimai atspindi skirtingas jos supratimo teorijas arba kriterijus žmogaus veiklai įvertinti. Antropologai ir kiti biheviorizmo mokslininkai dažnai apibrėžia kultūrą kaip visą spektrą žmogaus išmoktų elgesio šablonų. Kultūros aplinka teikia individui emocinę erdvę, kurioje įsitikinimų rinkinys, nuostatos ir elgesys gali būti būdingi visiems tam tikros socialinės arba etninės grupės nariams (Ember, Ember, 1977). Kultūra teikia kontekstą, kuriame suprantamas pasaulis, elgesio, bendravimo, sąveikos ir supratimo taisyklės (Evers, 2001b).

Vaske ir Grantham (Vaske, Grantam, 1990) pažymi, kad kultūra yra:

a) adaptuojama (turi savybę prisitaikyti prie fizinės ir socialinės aplinkos tam tikrų sąlygų);
b) integruojama (kultūrą sudarančios savybės paprastai yra suderinamos viena su kita);
c) nuolat keičiama (dėl adaptavimo prie tam tikrų kultūrinių įvykių arba integravimosi su kitomis kultūromis).

Nemažai autorių pažymi, kad egzistuoja ryšys tarp kultūros ir programinės įrangos tinkamumo naudoti (pvz., Qingxin Shi, 2007). Taip pat galima teigti, kad programinė įranga veikia kultūros raidą. Todėl programinės įrangos inžinierius turėtų suprasti kultūrines charakteristikas ir kultūrinius skirtumus tam, kad galėtų sukurti programinę įrangą, kurią būtų galima pritaikyti įvairioms kultūroms.

Pagrindiniai kultūros elementai, naudojami programinėje įrangoje, aptariami toliau.

4.2.1. Abėcėlės ir vardai

Programinė įranga operuoja įvairiais objektais, kurių vardus naudoja ne tik kompiuteriai, bet ir žmonės. Gimtąja kalba užrašyti vardai yra lengviau suprantami, įsimenami, manipuliuojami, taisomi, perteikiami bendraujant, o taip pat asocijuojami su ta kalba (Dürst, 2003). Kai kurioje senesnėje programinėje įrangoje egzistavęs reikalavimas varduose vartoti tik 26 raides (anglų k. abėcėlę) yra labai ribojantis net toms kalboms, kurios vartoja lotynų abėcėlę (nekalbant apie kitų raštų rašmenis), nes jų abėcėlės turi papildomų raidžių (pvz., å, š, ž, ...), o kai kurios anglų kalbos raidės nėra vartojamos (dažniausiai q, w, ir x).

Abonento vardai. Daugelyje programų (pvz., virtualiosiose mokymosi aplinkose, elektroninio pašto programose, pokalbių programose ir kt.) tikrinamas naudotojo tapatumas. Kaip vienas iš tokio tikrinimo komponentų paprastai pateikiamas abonento vardas, kuriuo naudotojas prisijungia prie sistemos. Deja, šiame varde daugelis sistemų leidžia vartoti tik pagrindinės lotynų abėcėlės raides, skaitmenis ir pabraukimo brūkšnį (_). Kai kurios sistemos (pvz., virtualioji mokymosi aplinka „ATutor“) abonento vardą naudoja ne tik vidiniam abonento identifikavimui sistemoje, bet ir kreipiniams į jį (jo vardas tampa matomas jam pačiam ir kitiems tos sistemos abonentams). Todėl suprantama, kad abonentas turėtų galimybę

pasirinkti vardą, sudarytą iš jo(s) gimtosios kalbos abėcėlės raidžių (ne tik iš ASCII ženklų). Nepaisant to, kad techninių kliūčių naudoti įvairių kalbų rašmenis abonentų varduose nėra, programų projektuotojai vis dar vengia realizuoti šią savybę savo produktuose. Dėl to lokalizuoti produktai nebeatitinka natūralumo reikalavimo toje kultūrinėje terpėje, kuriai jie lokalizuojami.

Asmenvardžiai. Daugelyje programų egzistuoja naudotojo duomenų (profilio) sritis, kur įrašomi ir laikomi asmeniniai duomenys. Suprantama, kad visos asmens tikrojo vardo ir pavardės (asmenvardžio) raidės turėtų būti priimamos (žmogus neturėtų keisti savo vardo ar pavardės rašybos tam, kad užsiregistruotų kurioje nors sistemoje).

Virtualiųjų mokymosi aplinkų analizė lokalizavimo požiūriu (Jevsikova, 2006) parodė, kad nacionalinių ženklų vartojimas asmenvardyje priklauso nuo serverio nuostatų: jei nuostatos teisingai parinktos lokalės atžvilgiu, tai visi tos lokalės kalbos ženklai gali būti vartojami. Analogiška situacija yra ir su kitomis dabartinėmis internetinėmis programomis, t. y. teisingai suderinus serverį ar operacinę sistemą, tokius ženklus galima surinkti asmenvardyje. Tačiau tenka pastebėti, kad nemažai žmonių vengia vartoti savo asmenvardžiuose kai kurias raides su diakritiniais ženklais, t. y. rašo savo asmenvardžius su rašybos klaidomis. Pokalbių programos „Skype“ abonentų vardų tyrimai rodo, kad teisingai užrašyti vardai ir pavardės (vartojant raides su diakritiniais ženklais) svyruoja nuo 15 proc. iki 96 proc., priklausomai nuo kalbos. Tirta vokiečių, čekų, danų, estų, islandų, latvių, lenkų ir lietuvių kalba užrašyti vardai (Grigas, Pedzevičienė, 2007). Toks aukštas „neraštingumo“ lygis gali būti paaiškintas įpročiu, perimtu iš kitų programų, kuriose dar yra ribojimų abonento vardų ženklams.

Slaptažodžiai. Slaptažodis – tai dar vienas naudotojo tapatumo tikrinimo komponentas daugelyje internetinių programų. Jis paprastai sudaromas iš raidžių ir skaitmenų. Tačiau daugelis sistemų susiaurina raidžių aibę iki ASCII ženklų, o tai nėra natūralu lokalėms, kurių kalbose vartojamos kitos abėcėlės. Be to, ženklų aibės susiaurinimas sumažina slaptažodžio saugumą.

Failų vardai. Operacinėse sistemose leidžiama vadinti failus tikrais vardais. Vartojamos raidės, skaitmenys ir kai kurie kiti teksto ženklai, pavyzdžiui, taškas, brūkšnelis. Neleidžiami tik ženklai, turintys specialią paskirtį šiame kontekste, pavyzdžiui, pasvirieji brūkšniai (jie vartojami keliui iki failo nurodyti), kablelis ir kt. Žmogus gali parinkti failų vardus savo kalba taip, kad vardas nusakytų failo turinį. Tas pats galioja ir aplankų (katalogų) vardams.

Galima išskirti kelis failų naudojimo atvejus:

1) dokumentų laikymas vietiniame kompiuteryje,
2) dokumentų mainai tarp kompiuterių naudojant keičiamąsias laikmenas,
3) dokumentų siuntimas kaip elektroninio laiško dalis ar priedas arba tiesiogiai, pvz., pokalbių programose,
4) failų, tinklalapių ar kitokio turinio laikymas serveryje,
5) naudojimas programos viduje.

1. Šiuo metu nėra kliūčių failus, laikomus vietiniame kompiuteryje, vadinti vardais, kuriuose yra nepagrindinės lotynų abėcėlės raidžių, jeigu operacinės sistemos lokalės nuostatos parinktos korektiškai. Galima kurti aplankų medį, kurio mazgų pavadinimai sudaromi iš gyvosios kalbos žodyno. Jei programa nepriima tokių failų, ją reikia laikyti pasenusia.
2. Dokumentų (failų) mainai tarp kompiuterių naudojant keičiamąsias laikmenas veikia gerai, jeigu ta pati 8 bitų koduotė naudojama abiejuose kompiuteriuose. Taigi mainai gali būti atliekami korektiškai nacionaliniame kontekste, taip pat gali būti naudojamos kaimyninių valstybių kalbos, naudojančios tą pačią koduotę. Unikodo realizacija pateiktų universalų sprendimą.
3. Dokumentų varduose, kurie yra siunčiami kaip elektroninių laiškų dalys arba priedai, arba naudojantis pokalbių programomis, ne ASCII ženklai yra vis dar retai vartojami dėl ribojimų, kurie anksčiau galiojo kai kuriose pašto programose, neteisingo pašto protokolų interpretavimo. Vardai prieš siunčiant failus yra koduojami UTF-8 kodų sekomis %FF, kuriose yra tik ASCII ženklai, o gavus – vardai iškoduojami. Siuntėjas ir gavėjas mato tik tikrus failų vardus. Metodas veikia tarptautiniu lygmeniu. Deja, ne visos programos priima vardus su ne ASCII ženklais.
4. Tinklalapių ir kito turinio laikymas serveryje, kai failų varduose pavartotos nacionalinės kalbos raidės, yra teoriškai išspęstas. Šiuo atveju vardai užkoduojami tokiu pačiu būdu, kaip elektroninių laiškų priedų vardai.
5. Failų ir aplankų (katalogų) vardai, kuriais operuoja programa ir kurie nėra matomi programos naudotojui, gali būti palikti neišversti. Tačiau būtų natūralu, kad visi vardai, kurie yra rodomi naudotojui, pvz., failų vardai, naudojami kaip elektroninio pašto aplankų vardai, turi būti lokalizuoti. Šiuo atveju reikalingas dinaminis failo vardo konvertavimas. Šio sprendimo realizacija – programos projektuotojo arba lokalizuotojo uždavinys.

Interneto sričių vardai. Naudotojas, dirbdamas su interneto programine įranga, dažnai operuoja sričių vardais. Ilgą laiką tik 26 raidės (anglų kalbos abėcėlė) galėjo būti vartojamos sudarant sričių vardus. 2003 metais buvo publikuotas dokumentas, reglamentuojantis tarptautinių ženklų naudojimą sričių varduose (Falstrom ir kt., 2003). Pagal šį dokumentą srities vardas (koduotas unikodu) yra konvertuojamas į ASCII ženklų eilutę, naudojant „Punycode“ kodavimą (tam, kad perduodami duomenys derėtų su esamomis sričių vardų sistemomis), o prieš rodant srities vardą naudotojui, jis yra iškoduojamas į unikodo ženklus. Naujausios versijos populiariausios naršyklės turi priemonių darbui su internacionalizuotais sričių vardais. Tačiau daugelis organizacijų vis dar vengia registruoti ir naudoti tokius sričių vardus. Viena iš galimų priežasčių – apgaulės rizika naudojant homografinius ženklus iš skirtingų abėcėlių, kai metodas realizuojamas tarptautiniu mastu. Kita priežastis – ne visos senesnės programos korektiškai veikia su internacionalizuotais sričių vardais, naudojamais saitais (hipernuorodomis).

4.2.2. Asmens vardo formatas

Asmenų vardai ir pavardės yra vartojami įvairiai, skiriasi jų pateikimo tvarka įvairiose kultūrose (pvz., „Vardas Pavardė“, „Pavardė Vardas“, „Pavardė, Vardas“). Internetinėje programinėje įrangoje vardai paprastai naudojami pasveikinant prisijungusį naudotoją, pradedantį darbo seansą, taip pat kaip išorinis naudotojo identifikavimo būdas, kuris gali būti matomas kitiems naudotojams (pvz., virtualiojoje mokymosi aplinkoje, turinio valdymo sistemoje, socialiniame tinkle). Todėl vardo ir pavardės tvarka turėtų būti įtraukta į lokalizuojamuosius išteklius, pvz., kaip atskiras parametras arba vardui ir pavardei vartojant skirtingus parametrų (kintamųjų) vardus, tam, kad juos būtų galima sukeisti vietomis.

4.2.3. Gramatinės formos

Daugelyje kalbų (pvz., lietuvių, suomių, lenkų ir kt.) vardai įvairiuose programos dialogo languose gali turėti įvairias formas. Į originalią programinę įrangą praktiškai neįmanoma įdėti visų kalbų visų linksnių generavimo modulį, tačiau galimybė prijungti tam tikros kalbos vardų generavimo modulį galėtų būti numatoma.

21 lentelė. Formų skaičius įvairiose kalbose

Formų skaičius	Kalbų skaičius	Kalbos
1	21	armėnų, gruzinų, indoneziečių, khmerų, korėjiečių, laosiečių, malajiečių, persų, tibetiečių, turkų ir kt.
2	73	anglų, baskų, bengalų, bulgarų, danų, estų, farerų, fryzų, hebrajų, hindi, mongolų, norvegų, suomių, vengrų, vokiečių ir kt.
3	14	baltarusių, bosnių, čekų, kašubų, kroatų, lietuvių, latvių, lenkų, makedonečių, rumunų, rusų, slovakų, serbų, ukrainiečių
4	4	kornų, maltiečių, slovėnų, sorbių, velsiečių
5	1	airių (Airijos gėlų)
6	1	arabų

Pastaba: duomenys paimti iš svetainių „Localization and Plurals“[1] ir „Translation Toolkits and Pootle“[2].

Pavyzdžiui, lietuvių kalboje šauksmininko linksnio generavimas reikalingas praktiškai visose internetinėse programose, kuriose vyksta dialogas su naudotoju, nes į jį reikia kreiptis šauksmininku. Tai gali būti asmens vardas, pavardė ar abiejų kombinacija. Nei viena iš analizuotų virtualiųjų mokymosi aplinkų (Jevsikova , 2006) tokios galimybės dar neturi. Tuo tarpu kai kurios bankinės ir kitos su verslu susijusios sistemos jau linksniuoja lietuviškus asmenvardžius.

Programinė įranga dažnai turi dialogo su naudotoju ekranus, kuriuose šalia skaičiaus (įvesto naudotojo ar sugeneruoto programos) rodomas atitinkamo objekto pavadinimas. Tačiau programa retai kada parenka teisingą objekto vardo formą, kuri atitiktų šalia esantį skaičių, pavyzdžiui, „1 objektas“, „9 objektai“, „11 objektų“. Anglų kalboje vartojamos dvi daiktavardžio formos (vienaskaita ir daugiskaita), tuo tarpu kitose kalbose vartojama daugiau formų (lietuvių, rusų, lenkų ir kt. kalbose yra trys formos: viena vienaskaitos, dvi daugiskaitos). Yra kalbų, kurios su bet kuriuo skaičiumi vartoja vieną ir tą pačią formą, bet yra turinčių net šešias skirtingas formas (21 lentelė).

Net ir vienodą formų skaičių turinčiose kalbose daiktavardžių formos su skaičiais derinamos nevienodai (skaičiai nevienodai skirstomi į grupes). Kalbos, turinčios tris formas (šiai grupei priklauso ir lietuvių kalba) pagal formos nustatymo algoritmą pasiskirsto į 7 grupes:

1) baltarusių, bosnių, kroatų, rusų, serbų, ukrainiečių;
2) čekų, slovakų;

[1] Localization and Plurals, https://developer.mozilla.org/En/Localization_and_Plurals

[2] Translation Toolkits and Pootle, http://translate.sourceforge.net/wiki/l10n/pluralforms

3) kašubų;
4) latvių;
5) lenkų;
6) lietuvių;
7) rumunų.

Gausiausia slavų kalbos grupė nuo lietuvių kalbos skiriasi tuo, kad skaičiai, kurie baigiasi 5, 6, 7, 8, 9 (išskyrus 15, 16, 17, 18, 19) priklauso ne antrai, o trečiai formai.

Latvių kalboje antrą skaičių grupę sudaro lietuviškų 2 ir 3 grupių junginys, išskyrus nulį, o į trečiąją grupę patenka tik vienas skaičius – nulis. Lietuvių ir daugelyje kitų kalbų su nuliu vartojama daugiskaitos formos (nulis sveikų, nulis daiktų), nors, formaliai žiūrint, tai nelogiška: mažiau už vienetą, o daugiskaita!

Daugiau formų turinčiose kalbose dažniausiai vartojama dviskaita. Kadaise ji buvo ir lietuvių kalboje (2 daiktu), bet dabar jau išnyko. Airijos gėlų kalboje po dviskaitos dar eina dvi formos skaičių grupėms 3–6 ir 7–10.

Arabų kalboje skaičiai į šešias grupes skirstomi šitaip:
1) 0;
2) 1;
3) 2;
4) baigiasi 03...10;
5) baigiasi 11...99;
6) visi kiti.

Straipsnyje (Boitet, 2005) pateikiamas formų derinimo ir eilučių kintamųjų reikšmių parinkimo būdas taikant baigtinius automatus.

4.2.4. Datos ir laiko formatai

Data gali būti pateikiama trumpu arba ilgu formatu. Ilgas datos užrašas (pvz., 2010 m. liepos 10 d.) yra sudėtingesnis, kadangi jame metus ir dienas atitinkantys skaičiai pateikiami kartu su teksto eilutėmis, kurių forma ir linksniai gali kisti. Trumpas datos užrašas taip pat yra skirtingas įvairiose kultūrose. Pavyzdžiui, Lietuvoje trumpas datos užrašas sudaromas iš trijų dalių. Pirmoji iš kairės – 4 skaitmenys, nusakantys metus, antroji – du skaitmenys, nusakantys mėnesį, ir trečioji – du skaitmenys, reiškiantys dieną. Jei mėnesio arba dienos numeris yra skaičius, mažesnis už 10, tai kairiau nuo reikšminių vienetų skaitmenų prirašoma po nereikšminį nulį. Lietuvoje trumpo datos užrašo dalys skiriamos brūkšneliais (pagal tarptautinį standartą

ISO 8601 ir Lietuvos standartą LST ISO 8601) arba tarpais. Pirmenybė teikiama minėtais standartais nusakytam skirtukui (brūkšneliui), pavyzdžiui, 2010-07-10. Kitose lokalėse naudojama kitokia datos komponentų tvarka (pvz., diena, mėnuo, metai arba mėnuo, diena, metai) ir skirtukai (pvz., /).

Lokalizuotinoje programinėje įrangoje turėtų būti galimybė rodyti datą pagal tarptautinį standartą (ISO 8601) arba nurodyti lokalizuojamuose ištekliuose pageidautiną formatą, datos komponentų (metų, mėnesio ir dienos) skirtukus ir eilę.

Virtualiųjų mokymosi aplinkų analizė lokalizavimo požiūriu (Jevsikova, 2006) parodė, kad dažnai vartojami sutrumpinti datos komponentų pavadinimai, pavyzdžiui, mėnesių santrumpos „Aug“, „Oct“ ir kt. Ne visose kalbose analogiškos santrumpos yra priimtinos.

Laiko matavimas gali būti priskirtas prie tarptautinių kultūros elementų, tačiau jo pateikimo formatas gali būti tarptautinis arba nacionalinis. Tarptautinis laiko formatas yra apibrėžtas tarptautiniu standartu ISO 8601, kuris įteisintas ir kaip Lietuvos standartas LST ISO 8601.

Laikas gali būti pateikiamas 12 ar 24 valandų formatu su įvairiais valandų ir minučių skirtukais (pvz., taškas, dvitaškis, tarpas). Pavyzdžiui, Didžiojoje Britanijoje naudojamas 12 valandų formatas (pvz., 9:30 AM, 5:45 PM), o daugelyje kitų Europos lokalių (pvz., Lietuvoje) naudojamas 24 valandų formatas (pvz., 17:45).

Laiko pateikimas programinėje įrangoje, veikiančioje serveryje (pvz., virtualiojoje mokymosi aplinkoje, turinio valdymo sistemoje) dažniausiai priklauso nuo serverio nuostatų ir pačios programinės įrangos nuostatų.

Dar reikėtų atkreipti dėmesį į pirmosios savaitės dienos nustatymą. Įvairiose lokalėse yra skirtingi susitarimai dėl pirmosios savaitės dienos. Pavyzdžiui, JAV – tai sekmadienis, tuo tarpu Lietuvoje, Lenkijoje, Rusijoje ir kt. šalyse – pirmadienis.

4.2.5. Matavimo vienetai

Lietuvoje (ir Europoje) vartojami metrinės sistemos vienetai. Didžiojoje Britanijoje ir JAV dar tebevartojami colinės sistemos ir kitokie nesisteminiai vienetai (mylia, svaras ir kt.).

Dažniausiai pasitaiko coliai (1” = 2,54 cm; paprastųjų kabučių ženklu anglai žymi colius). Jais matuojamos teksto eilučių įtraukos, lapo paraštės, ekranų įstrižainės ir kt. Todėl programinėje įrangoje paprastai būna numatyta galimybė matavimo vienetus pasirinkti. Lokalizuojant programą tokia galimybė dažniausiai paliekama,

bet lietuviškoje lokalizacijoje numatytoji vienetų sistema turi būti metrinė. Matavimo vienetų pavadinimai verčiami bet kuriuo atveju.

Kol kas skiriamoji geba matuojama taškais (pikseliais) colyje. Čia coliai taip įsigalėję, kad dar tenka taikstytis su jais.

Būna atvejų, kai anglų kalboje pavartotas matavimo vienetas yra nevienareikšmis. Anglai popieriaus svorį išreiškia 500 (kartais 1000) nesupjaustytų lapų svoriu svarais, o mes jį išreiškiame metrinės sistemos vienetais – kvadratinio metro svoriu gramais: g/m^2. Angliškoje sistemoje lapų dydžių yra keletas (priklauso nuo popieriaus paskirties). Rašomojo popieriaus nesupjaustyti lapai paprastai būna 17×22 colių dydžio, o pake būna 500 lapų. Tokio popieriaus 20 lb (20 svarų) storis išreiškus mūsuose vartojamais vienetais būtų 75,2 g/m^2. Tačiau kitokios paskirties popieriaus lapų dydis gali būti kitoks (pvz., Vikipedijoje[1] pateikiama net dešimt popieriaus formatų). Lokalizuotojas turėtų išsiaiškinti, kokio dydžio lapams nurodytas svoris ir jį perskaičiuoti į kvadratinio metro gramus.

Metrinės matavimo sistemos vienetai turi tarptautinius žymenis (santrumpas). Štai keletas dažniau vartojamų: *m – metras, cm – centimetras, mm – milimetras, Hz – hercas, kHz – kilohercas, MHz – megahercas, g – gramas, kg – kilogramas, h – valanda, min – minutė, s – sekundė*.

Po tarptautinės matavimo vieneto santrumpos taškas nededamas (plg., *m – metras, m. – metai*).

Laiko vienetams dažniau vartojamos lietuviškos santrumpos: *val. – valanda, min. – minutė, sek. – sekundė*. Kitiems vienetams – tarptautinės.

4.2.6. Dešimtainės trupmenos ir tūkstančių skirtukai

Tarptautiniame standarte ISO 31-1 apibrėžti du dešimtainės trupmenos skirtukų tipai: kablelis ir taškas. JAV, Jungtinėje Karalystėje, daugelyje kitų šalių, ypač tų, kuriose vartojama anglų kalba, dešimtainės trupmenos skirtukas yra taškas, Vokietijoje, Prancūzijoje, daugelyje kitų Europos ir Pietų Amerikos šalių – kablelis. Neteisingas dešimtainės trupmenos skirtukas tam tikroje lokalėje gali būti supainiotas su tūkstančių skirtuku ir tapti klaidingų skaičių reikšmių suvokimo priežastimi. Kai kurios programos automatiškai ima skirtuko tipą iš operacinės sistemos lokalės nuostatų. Jei ne, tai skirtuko tipą turėtų būti galima pasirinkti programos lokalizavimo metu. Deja,

[1] Wikipedia, http://en.wikipedia.org/wiki/Paper density.

yra programų, kuriose šis skirtukas užkoduotas programos viduje ir jį labai sunku pakeisti – reikia papildomų programavimo pastangų lokalizavimo metu.

JAV naudojamas tūkstančių skirtukas kablelis, o daugelyje Europos šalių – taškas arba tarpas. Neteisingas tūkstančių skirtukas gali būti supainiotas su dešimtainės trupmenos skirtuku ir tapti klaidingų skaičių reikšmių suvokimo priežastimi.

4.2.7. Telefono numerio formatas

Telefono numerio formatas įvairiose šalyse gali būti skirtingas. Programinėje įrangoje jį paprastai galima rasti naudotojo duomenų profilyje. Jei sritis telefono numeriui surinkti yra iš keleto langelių, tai lokalizavimo metu turėtų būti galimybė pakeisti langelių skaičių ir (arba) eilę. Vienas iš nesuderinamumų šalinimo sprendimų – naudoti vieną langelį telefono numeriui surinkti.

4.2.8. Komponuojamos ir parametrizuotos eilutės

Nemažai problemų lokalizavimo metu kelia sakiniai arba frazės, sukomponuotos iš keleto atskirų eilučių, o taip pat parametrai, įdėti į lokalizuotinas eilutes. Toks parametrizavimas gali tikti vienai kalbai, bet gali būti visiškai nepriimtinas kitai, nes eilučių eilė (tvarka), žodžių formos, didžiųjų ir mažųjų raidžių vartojimas eilutės viduje skiriasi įvairiose kalbose. Todėl parametrų naudojimas turėtų būtų pakomentuotas ištekliuose, kad lokalizuotojas galėtų įdėti parametrą į tinkamą eilutės vietą, arba ieškoma naujų sprendimų, kaip pateikti informaciją apie parametrizavimo ir komponavimo atvejus ir kontekstą, kuriame bus naudojama visa eilutė.

4.2.9. Rašybos tikrinimas, skiemenavimas

Rašybos tikrinimas yra būtina bet kokios programos, kuria rengiami tekstai, funkcija. Tai raštinės paketai, elektroninio pašto ir pokalbių programos, interneto naršyklės (jose pildomos anketos, rašomi tinklaraščiai, komentarai) ir kitos programos.

Įvairių kalbų rašybos taisyklės skirtingos, jose būna daug išimčių. Dėl to sunku padaryti gerus rašybos tikrinimo komponentus. Pastaraisiais metais populiariais tapo atvirieji rašybos tikrinimo komponentai, kurių kokybė nuolat gerėja ir kuriuos galima laisvai pritaikyti specifiniams kultūriniams ir programiniams poreikiams.

Skiemenavimas susijęs su eilučių laužymu. Ieškoma optimalios vietos, ties kuria gali būti laužiama eilutė. Skiemenavimas taip pat yra susijęs su rašybos tikrinimu dėl panašių gramatinės ir morfologinės analizės algoritmų. Eilutės laužimo vieta aptinkama dviem etapais: 1) identifikuojama galima eilutės laužimo vieta (ji gali būti žymima valdymo ženklais, nurodančiais galimą eilutės laužimo vietą (pvz., U+000D, U+2028 ir kt.) arba vieta, kurioje negalima laužti eilutės (pvz., U+00A0)); 2) teksto formatavimo algoritmas išrenka optimalią eilutės laužimo vietą. Skirtingoje programinėje įrangoje skiemenavimas realizuotas taikant įvairius algoritmus. Unikodo standartas teikia bendro algoritmo, tinkamo įvairiems eilučių laužymo atvejams, specifikaciją.

4.3. Lokalių modeliai

Lokalėse apibrėžiami kultūros elementai ir pateikiamos priemonės šiems elementams formaliai aprašyti. Šiame skyrelyje analizuojami svarbiausi dokumentai ir projektai, skirti lokalės duomenų aprašams apibrėžti. Dalis šių dokumentų yra tarptautiniai standartai arba specifikacijos.

Toliau nagrinėjami pagrindiniai lokalių modeliai, kuriais naudojamasi projektuojant internacionalizuotą programinę įrangą: nuo programavimo kalbos nepriklausomos POSIX ir FDCC lokalės, Javos lokalė (kadangi nemažai internetinių programų projektuojama remiantis Javos technologijomis), tarptautinio standarto ISO/IEC 15897 apibrėžiamas lokalių modelis (kultūrinių elementų registravimo procedūra) ir CLDR lokalė kaip pagrindas didžiausiai šiuo metu egzistuojančiai lokalių duomenų bazei kurti. Esama ir kitų lokalių modelių – tai įvairių programavimo kalbų ir programavimo priemonių naudojamos lokalės, kurių atskirai nenagrinėsime dėl jų struktūros panašumo į čia analizuojamas pagrindines lokales.

4.3.1. POSIX lokalė

POSIX lokalė – viena pirmųjų priemonių lokalės formaliam aprašui suvienodinti. Ji buvo įtraukta į ISO/IEC 9945-1 ir ISO/IEC 9945-2 standartus. Pirmoji standarto versija pasirodė 1988 metais. Su lokale susijusios pirmos dvi standarto dalys:

22 lentelė. POSIX lokalės kategorijos

Kategorija	Aprašas
LC_CTYPE	Ženklų klasifikacija (mažosios ir didžiosios raidės, dešimtainiai ir šešioliktainiai skaitmenys, skyrybos ženklai, tarpai, spausdinami ženklai ir kt.), ženklų transformacijos ir kiti ženklų požymiai
LC_COLLATE	Ženklų rikiavimo taisyklės (ženklų sekų rikiavimas, rikiavimo svoriai, ženklų susiejimas, ženklų ekvivalenčių klasių apibrėžimas ir kt.)
LC_MONETARY	Pinigų sumų užrašymo formatai (valiutų simboliai), dešimtainės trupmenos ir tūkstančių skirtukai pinigų sumose, skaitmenų grupavimas, teigiamos ir neigiamos reikšmės, valiutos ženklo vieta (prieš arba po sumos) ir kt.
LC_NUMERIC	Skaičių rašymo formatai (dešimtainės trupmenos sveikosios ir trupmeninės dalies skirtukas, tūkstančių skirtukas, skaitmenų grupavimo taisyklės)
LC_TIME	Datos ir laiko formatai (savaitės ir mėnesių sutrumpinti ir nesutrumpinti pavadinimai, datos ir laiko komponentų tvarka, 12 ar 24 val. laiko formatas, eros pavadinimas ir kt.)
LC_MESSAGES	Informatyvių pranešimų formatai (teigiami ir neigiami atsakymai)

POSIX.1 (atitinka ISO/IEC 9945-1) ir POSIX.2 (ISO/IEC 9945-2), apibrėžiančios portatyvių programų sąsają, komandas ir priemones. Naujausios šių standartų versijos buvo priimtos 2003 metais.

POSIX lokalės modelis sudaro programavimo kalbos C lokalės pagrindą, kuris yra apibrėžtas ISO/IEC 9899 standarte. Jame pateiktos formalios taisyklės kiekvienam kategorijos elementui aprašyti. Naudoja įvairiose sistemose veikiantys C kompiliatoriai. POSIX lokalės modelis taip pat naudojamas „Unix“ sistemose. „Windows“ operacinės sistemos turi su POSIX lokale suderinamą posistemį.

POSIX lokalėje naudojamos kategorijos pateiktos 22 lentelėje.

POSIX lokalių modelio privalumai:

1. Tai tarptautinis standartas, kuris pirmasis formaliai apibrėžė svarbiausius lokalės elementus ir su kuriuo suderinama nemažai projektuojamos programinės įrangos.
2. Portatyvumas: nepriklausomumas nuo operacinės sistemos ar kt. platformos.

Tačiau yra ir trūkumų:

1. Kategorijos negali būti išplėstos (arba kopijuojamos, arba aprašomos iš naujo).
2. Galimas nesuderinamumas su koduotėmis (nėra natūralios galimybės nurodyti unikodo kodus).

3. Per mažai kategorijų. Jos neapima pagrindinių su kalba ir vietove susijusių kultūrinių normų.

4.3.2. FDCC lokalė

Šis lokalės modelis pateiktas tarptautiniame standarte ISO/IEC 14652 (kultūrinių susitarimų formalių apibrėžčių rinkinys). Tai POSIX ir C lokalės, apžvelgtos aukščiau, viršaibis, skirtas panaikinti POSIX lokalės modelio trūkumus. Pirmosios šešios kategorijos yra POSIX lokalės kategorijos, kitos – naujos kategorijos (23 lentelė).

Be to, POSIX kategorijos standarte ISO/IEC 14652 yra išplėstos, pavyzdžiui, pinigų sumų kategorijoje yra galimybė nurodyti, ar yra tarpas tarp sumos ir valiutos ženklo, ar naudojami kiti ženklai valiutos ženklui atskirti. Skaičių formatų kategorija papildyta galimybe užrašyti sudėtingesnes skaitmenų grupavimo taisykles. Datos ir laiko kategorijoje įvesta galimybė aprašyti savaites, kuriose yra ne septynios dienos (pvz., Vakarų Afrikos akanų kai kurių bendruomenių tebenaudojama 6 dienų savaičių sistema). Ženklų rikiavimo kategorijoje įtraukti efektyvesni rikiavimo būdai, įvestos ženklų transliteracijos taisyklės. Ženklai gali būti nurodomi unikodo kodų kaitos sekomis <Uxxxx> arba <Uxxxxxx>.

Standartas patobulino POSIX lokalės modelį: atsirado svarbių naujų kategorijų, tačiau jis nėra visiškai suderinamas su POSIX programomis.

23 lentelė. FDCC lokalės kategorijos

Kategorija	Aprašas
<POSIX kategorijos>	Žr. 22 lentelę
LC_IDENTIFICATION	Metainformacija apie lokalę: kūrėjas, lokalės kodas
LC_ADDRESS	Pašto adresų užrašymo taisyklės, valstybių pavadinimai, priimtos valstybių pavadinimų santrumpos, naudojamos įvairiose srityse, pvz., ant transporto priemonių, ISBN numerių, kalbų pavadinimai, kalbų pavadinimų santrumpos ir kt.
LC_MEASUREMENT	Informacija apie naudojamą matavimo sistemą
LC_NAME	Asmenvardžių ir titulų rašymo taisyklės, kreipiniai į vyrus ir moteris drauge, į vyrą, netekėjusią ir ištekėjusią moteris, universalus kreipinys į moteris
LC_PAPER	Dokumentams naudojamas numatytasis popieriaus lapo dydis
LC_TELEPHONE	Telefonijos paslaugų naudojami formatai: tarptautinis formatas, lokalus formatas, kodų prefiksai

4.3.3. Javos lokalė

Javos programavimo kalba ir programavimo terpė naudoja savą objektinį lokalės modelį, apibrėžtą Javos kalbos specifikacijoje. Lokalės kategorijas atitinka Javos klasės, turinčios įvairių metodų darbui su lokalės elementais. Naudojami atvirieji, laisvai platinami lokalių aprašai (aprašyta daugiau kaip 100 lokalių), kurių duomenys pasiekiami naudojantis Javos lokalės klasėmis ir metodais, įtrauktais į *java.text* ir *java.utils* pakuotes.

Javos lokalės modelį sudaro tokie lokalės elementai (Deitsch, Czarnecki, 2001): skaičių formatai (dešimtainės trupmenos skirtukai ir tūkstančių skirtukai), datos ir laiko formatai (trumpasis ir ilgasis), valiutų simboliai, kalendoriaus elementai, laiko juostos, rikiavimas ir kiti veiksmai su tekstu, grafinių komponentų išdėstymas (pvz., iš kairės į dešinę ar iš dešinės į kairę), lokalės išteklių (teksto, grafikos, garso, vaizdo įrašų ir kt.) atskyrimas. Klasės, skirtos išvardytoms lokalių kategorijoms, nurodytos 24 lentelėje.

Lokalizuojamiems ištekliams atskirti Javos modelyje naudojami išteklių rinkiniai. Tekstiniai ištekliai atskiriami įtraukiant juos į PROPERTIES failus, o nuo lokalės priklausomi multimedijos ištekliai atskiriami naudojant klasę *ListResourceBundle*.

Javos lokalės modelyje naudojama daugiau elementų, negu POSIX, tačiau mažiau, lyginant su FDCC, bet pateikiamos konkrečios darbo su lokalės elementais kal-

24 lentelė. Javos lokalės kategorijos

Kategorija	Klasės ir metodai
Lokalės informacija	Klasė `Locale`
Skaičių ir pinigų sumų formatai	Klasė `NumberFormat`, Metodai: `getNumberInstance, getPercentInstance, getCurrencyInstance, getIntegerInstance, Format`
Datos ir laiko formatai	Klasės: `DateFormat, SimpleDateFormat`, Metodai: `getDateInstance, getTimeInstance, getDateTimeInstance, Format`
Kalendorius	Klasės: `Calendar, GregorianCalendar`
Laiko juosta	Klasė: `TimeZone`
Rikiavimas ir kiti veiksmai su tekstu	Klasės: `Collator, RuleBasedCollator, BreakIterator` Metodai: `GetInstance, Compare, getWordIterator, getLineIterator, getSentenceIterator, getCharacterIterator` ir kt.
Grafinių elementų išdėstymas	Klasė: `ComponentOrientation`, Metodas: `GetOrientation`
Lokalės išteklių atskyrimas	Klasės `ResourceBundle, ListResourceBundle`

bos konstrukcijos, nuo pat pradžios įvestas unikodas. Kai kurias FDCC kategorijas, kurių nėra Javos modelyje, galima realizuoti kitomis priemonėmis, pvz., asmenvardžio užrašymo taisykles – per parametrizuotas eilutes.

4.3.4. ISO/IEC 15897 lokalė

Kultūros elementų registravimo standartas ISO/IEC 15897 apibrėžia kultūros elementų, pateiktų tiek nusakomuoju tekstu, tiek formaliai, registravimo procedūras. 2001 metais šis standartas buvo priimtas Lietuvoje (LST ISO/IEC 15897). Standartas suderinamas su ISO/IEC 9945-2 (POSIX) aprašytomis kategorijomis, tačiau leidžia registruoti daugiau lokalės elementų. Registravimo rezultatai yra laisvai prieinami: programinės įrangos gamintojai gali laisvai jais pasinaudoti.

Pagal šį standartą gali būti registruojami keturi kultūros specifikacijų tipai:

1. Nusakomoji kultūros specifikacija.
2. POSIX lokalė.
3. POSIX ženklų koduotė.
4. POSIX ženklų aibė.

Nusakomoji kultūros specifikacija apibrėžia kultūros normas pasakojamąja anglų kalba, kartu joje gali būti pateiktos ir ekvivalenčios apibrėžtys kitomis kalbomis. Ji taikytina tada, kai trūksta formaliais kultūros normų specifikavimo metodais paremtos kodifikacijos. Jei kai kurie nusakomosios kultūros specifikacijos elementai taip pat apibrėžiami POSIX lokalėje ar POSIX ženklų koduotėje, tai specifikacija turi nurodyti tą lokalę ar ženklų koduotę.

2, 3 ir 4 tipai yra aprašomi naudojant kultūros elementų POSIX specifikacijas, apibrėžtas ISO/IEC 9945-2 standarte.

Nusakomosios kultūros specifikacijos pirmieji šeši skyriai sutampa su POSIX lokalės kategorijomis, kiti skyriai skirti išsamesniam atitinkamuose pirmuose šešiuose POSIX skirsniuose išvardytų kultūros aspektų aprašymui (25 lentelė). Aprašymai pateikiami neformaliai, tačiau formatu, kurį galėtų apdoroti kompiuteris.

Šio standarto privalumai:

1. Suderinamas su POSIX lokalės modeliu.
2. Teikia priemonių aprašyti nemažai svarbių papildomų kultūros elementų, kurių nėra nei POSIX, nei FDCC, nei Javos lokalėse.

Nepaisant šių privalumų, lokalių duomenys nėra aktyviai registruojami, taigi programinės įrangos projektuotojai negali realiai jais pasinaudoti.

25 lentelė. Nusakomosios kultūros specifikacijos elementai

Specifikacijos skyrius	Aprašas
POSIX lokalės kategorijų skirsniai	Žr. 22 lentelę
Nacionalinė ir regioninė informacinių technologijų terminija	Gali būti išvardyta terminai bendrine kalba ir regioniniai jų variantai, pavyzdžiui, informacinių technologijų ISO terminų vertimai
Nacionaliniai ir regioniniai standartų profiliai	Gali būti išvardyti standartų profiliai, pavyzdžiui, atvirųjų sistemų sujungimo (OSI) nacionaliniai profiliai arba POSIX standartų profiliai
Ženklų rinkinio aptarimas	Aprašoma rašto ženklų vartosena, pavyzdžiui, kokie rašto ženklai būtinai reikalingi konkrečiai kalbai; kokie kiti ženklai vartojami kalbai niuansuoti; kokiais ženklais įprasta laikraščiuose ir knygose rašyti asmenvardžius bei vietovardžius; kokie ženklai vartojami senuosiuose raštuose; kitiems tikslams vartojami ženklai
Rikiavimo ir paieškos taisyklės	Taisyklės, kaip skaidyti įrašą į rikiavimo laukus, kurie specialūs žodžiai ignoruojami atliekant palyginimo ar paieškos veiksmus. Čia taip pat gali būti aprašytos palyginimo taisyklės, kuriose remiamasi žodžių tarimu
Ženklų transformacijos	Ženklų transliteracijos ir transkripcijos, pavyzdžiui, lotynų, graikų ir kirilicos rašmenų transliteracijos, arba kai kurių raidžių pakaitalai
Ženklų savybės	Papildoma informacija apie ženklų savybes, pvz., kaip mažosios raidės, neturinčios didžiųjų atitikmenų, turi būti rašomos ten, kur įprasta rašyti didžiąsias raides
Specialiųjų ženklų vartojimas	Kabučių, santrumpų ženklų bei skirtukų ir kitų ženklų vartosena
Ženklų vaizdų formavimas	Kokios ženklų vaizdavimo alternatyvos laikomos adekvačiomis ir kokie ženklų grafiniai vaizdai priimtini
Ženklų įvedimas	Įvedimas klaviatūra ir kiti ženklų įvedimo metodai
Asmenvardžių sudarymo taisyklės	Taisyklės, kas laikoma pavarde, kaip rašomas titulas, ar leistina pavardes rašyti vien didžiosiomis raidėmis, ar rašomi pilni vardai, ar inicialai. Čia taip pat galima pateikti taisykles, nurodančias, kaip vaikai paveldi tėvo ir motinos pavardę, kokios būna sutuoktinių pavardės
Kaityba	Pateikiamos kalbos kaitybos taisyklės arba nuorodos, kur jas rasti
Žodžių kėlimas	Žodžių kėlimo taisyklės arba kur jas galima rasti
Rašyba	Rašybos taisyklės ir rašybos žodynai bei nuorodos į ortografijos šaltinius
Numeravimas, kelintiniai skaitvardžiai bei matų sistemos	Matų sistemos (paprastai ISO SI)

25 lentelė (tęsinys)

Specifikacijos skyrius	Aprašas
Pinigų sumos	Papildoma POSIX kategorija, pvz., senoviški piniginiai vienetai
Data ir laikas	Papildoma atitinkama POSIX kategorija joje nenumatytais datų rašymo formatais, laiko juostų pavadinimais, vasaros laiko įvedimo taisyklėmis bei kitais rašytinėje kalboje vartojamais dalykais
Informacija apie valstybę	Įvairios žinios apie valstybę, pavyzdžiui, pašto indeksai, administracinio suskirstymo padalinių kodai, policijos skyrių kodai, teritorijų pavadinimų santrumpos
Telefonų numeriai	Vietinių ir tarptautinių telefonų numerių rašymo formatai
Pašto adresai	Pašto adresų rašymo forma: kur rašomas adresatas, gatvės pavadinimas ir pašto indeksas, pastato aukštų pavadinimai ir panaši informacija
Asmenų ir organizacijų identifikavimas	Identifikavimo sistemos, pavyzdžiui, socialinio draudimo numeriai, įmonių kodai ir kt.
Elektroninio pašto adresai	Elektroninio pašto adresų reglamentacija
Sąskaitų numeriai	Valstybei būdinga banko sąskaitų numerių sandara
Ženklų išdėstymas klaviatūroje	Aprašomas ženklų išdėstymas klaviatūroje
Žmogaus ir kompiuterio dialogas	Gali būti aprašyta, kaip lokalizuoti programas
Popieriaus lapo formatai	Popieriaus lapo formatai (paprastai ISO standartai), vokų su langeliais naudojimas ir pan.
Maketavimo dalykai	Aprašoma, kaip maketuoti, pavyzdžiui, verslo laišką arba faksogramą

4.3.5. CLDR lokalė

CLDR projektas pradėtas 2004 metais, tęsiant 2003 metais „OpenI18N“ grupės pradėtą kurti XML lokalės duomenų saugyklą. Projektą vykdo „Unicode“ konsorciumas[1]. CLDR tikslai: pateikti priemones bendriems programinėje įrangoje naudojamiems įvairių pasaulio lokalių duomenims specifikuoti, sukaupti kuo daugiau lokalių duomenų (26 lentelė). Lokalių duomenų mainams naudojamas XML formatas (lokalės duomenų žymėjimo kalba LDML) (Davis, 2007), o duomenys laisvai prieinami internete. Naudojamas duomenų kodavimo būdas – UTF-8.

[1] Unicode, Inc. Unicode CLDR Project: Common Locale Data Repository, 2007. http://unicode.org/cldr/ [žiūrėta 2010-06-29].

26 lentelė. LDML lokalės kategorijos

LDML kategorijos	Lokalės elementas
<identity>	Informacija apie lokalę ir lokalių identifikatoriai
<localeDisplayNames>	Kalbų, kalbų grupių, valstybių, teritorinių vienetų, kalendorių, laiko juostų, valiutų pavadinimų vertimas į lokalės kalbą
<layout>	Teksto kryptis, programų grafinės sąsajos elementų išdėstymas
<characters>	Lokalės kalboje vartojami ženklai ir jų klasifikacija (pagrindiniai kalbos ir papildomi ženklai)
<delimiters>	Kabutės, įdėtinės kabutės
<measurement>	Matavimo sistema, dokumentų lapų dydžiai
<dates>	Datos ir laiko ilgieji ir trumpieji formatai (įskaitant savaitės dienų pavadinimus, šablonus). Naudojamas kalendorius, erų pavadinimai, mėnesių pavadinimai ir jų santrumpos, metų ketvirčių ar jų ekvivalentų pavadinimai, savaitės dienų pavadinimai ir jų santrumpos, savaitės pirmoji diena, laiko juostos. Datos formatai remiasi Javos lokalės modelyje naudojamais formatais
<numbers>	Skaičių ir pinigų sumų formatai (skaitmenų grupavimas, skirtukai, teigiamos ir neigiamos sumos, valiutos simbolis ir jo vieta ir kt.)
<posix>	Ši kategorija įtraukta suderinamumui su POSIX lokale užtikrinti, naudojamos POSIX lokalės kategorijos
<collations>	Ženklų rikiavimas, paieška, lyginimas ir transformacijos

Tai didžiausia šiuo metu egzistuojanti lokalės duomenų bazė (2009 m. pateikta daugiau kaip 100 lokalių formalių aprašų). CLDR projektas taip pat teikia priemones duomenims eksportuoti į su POSIX suderinamą formatą, Javos naudojamus išteklių rinkinius, raštinės paketo „OpenOffice.org“ naudojamą lokalės formatą. Nuo 2006 metų CLDR lokalė naudojama „Mac OS X“, „Solaris“, „AIX“ operacinėse sistemose, „Acrobat“, „ModernBill“ programinėje įrangoje, „OpenOffice.org“ raštinės programų pakete (Moore, 2006).

4.3.6. Lokalių modelių kultūros elementų palyginimas

Pateiksime ankstesniuose skyreliuose analizuotų lokalių modelių lyginamąją lentelę (27 lentelė). Lokales lyginsime pagal tai, ar atitinkamas lokalės modelis turi kairiajame lentelės stulpelyje pateiktus lokalių elementus (žymima +), ar jų neturi (žymima –). Naudosime iš visų lokalių sukomponuotą elementų grupių sąrašą.

27 lentelė. Analizuotų lokalių modelių palyginimas

Lokalės elementai	POSIX	FDCC	Java	ISO/IEC 15897	CLDR
Ženklai, jų klasifikacija ir transformacijos	+	+	+	+	+
Ženklų rikiavimas ir paieška	+	+	+	+	+
Pinigų sumų užrašymo formatai	+	+	+	+	+
Skaičių rašymo formatai	+	+	+	+	+
Datos ir laiko formatai, savaičių, mėnesių, erų pavadinimai	+	+	+	+	+
Laiko juostos ir jų pavadinimai	–	–	+	+	+
Teigiami ir neigiami atsakymai	+	+	+	+	+
Pašto adresai, valstybių ir kalbų pavadinimai	–	+	+	+	+
Matavimo vienetai	–	+	+	+	+
Asmenvardžių, titulų, kreipinių užrašymo taisyklės	–	–	–	+	+
Popieriaus formatai	–	+	–	+	+
Telefonų numerių ir kodų formatai	–	+	–	+	+
Grafinių elementų išdėstymo kryptis	–	–	+	+	+
Informacinių technologijų terminija	–	–	–	+	–
Nacionaliniai standartai	–	–	–	+	–
Specialiųjų ženklų ir skirtukų vartojimas	–	–	–	+	+
Ženklų vaizdų formavimas	–	–	–	+	–
Ženklų įvedimas, klaviatūros, išdėstymas klaviatūroje	–	–	–	+	–
Kaityba ir rašyba (nuorodos į taisykles)	–	–	–	+	–
Informacija apie valstybę	–	–	–	+	–
Elektroninio pašto adresai	–	–	–	+	–
Banko sąskaitų numeriai	–	–	–	+	–
Maketavimo ypatumai	–	–	–	+	–
Lokalės išteklių atskyrimo priemonės	–	–	+	–	–
Iš viso	6	10	11	23	14

ISO/IEC 15897 standartas numato priemones didžiausiam lokalių elementų grupių skaičiui apibrėžti lokalėje, tačiau nėra plačiai vartojamas. CLDR lokalė numato priemones pateikti 14 iš lentelėje išskirtų 24 lokalės elementų grupių informacijai ir turi sukauptų ir laisvai prieinamų lokalių aprašų.

Viena iš lokalizavimo problemų priežasčių yra ta, kad internacionalizuojant programinę įrangą naudojamasi esamais lokalių modeliais. Pagrindinių lokalių modelių analizė parodė, kad kiekvienas istoriškai vėliau atsirandantis lokalės modelis įtraukia daugiau lokalės elementų ir būna skirtas pagerinti ankstesnius žinomus modelius, tačiau:

1) skirtingi lokalių modeliai nėra tarpusavyje suderinami (paprastai išlaikomas tik naujo modelio suderinamumas su POSIX lokale, kurios kategorijų nepakanka, norint tinkamai lokalizuoti programinę įrangą);

2) nei vienas lokalės modelis neapima pakankamai lokalių kultūrinių normų (pvz., giminių, linksnių, mažųjų ir didžiųjų raidžių derinimo ir frazių komponavimo), dėl to reikia ieškoti kitų būdų problemoms, susijusioms su minėtomis kultūrinėmis normomis, spręsti.

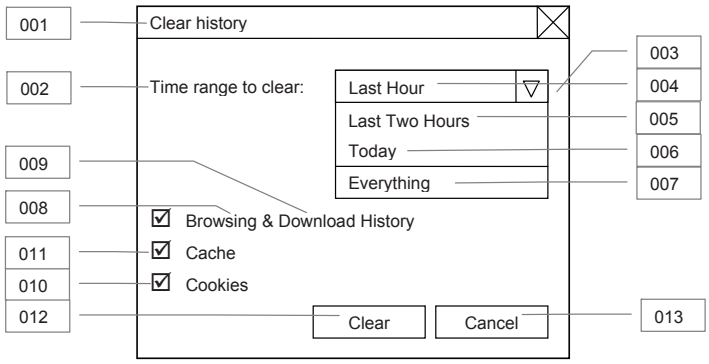
5. LOKALIZUOJAMIEJI IŠTEKLIAI

Planuojamos lokalizuoti programinės įrangos kūrėjai turi ją internacionalizuoti. Vienas svarbiausių šio darbo etapų – lokalizuojamųjų išteklių atskyrimas nuo pirminių programos tekstų. Tai visų tekstų, grafikos, garsų, lokalės elementų, pagalbinių parametrų ir kitų elementų, pateikiamų naudotojui veikiant programai, atskyrimas nuo programos vykdomosios dalies – iškėlimas į atskirus failus arba atskiras vykdomųjų failų sekcijas (išteklių sekcijas).

Lokalizuojamieji programos ištekliai gali būti tekstiniai ir dvejetainiai. Jų pateikimas priklauso nuo programavimo kalbos, kuria parašyta programinė įranga, naudojamo kompiliatoriaus, platformos, kuriai kuriama programinė įranga, taip pat naudoto išteklių atskyrimo metodo. Kai kurie lokalizuojamųjų išteklių atskyrimo metodai pateikia tik tekstinius išteklius. Nuorodos į dvejetainius objektus (pvz., grafiką, vaizdo, garso įrašus) gali būti pateikiamos tekstiniu pavidalu – jų universaliuoju identifikatoriumi vidinėje arba tinklo išorinėje lokalizuojamųjų išteklių sistemoje.

5.1. Lokalizuojamųjų išteklių pavyzdys

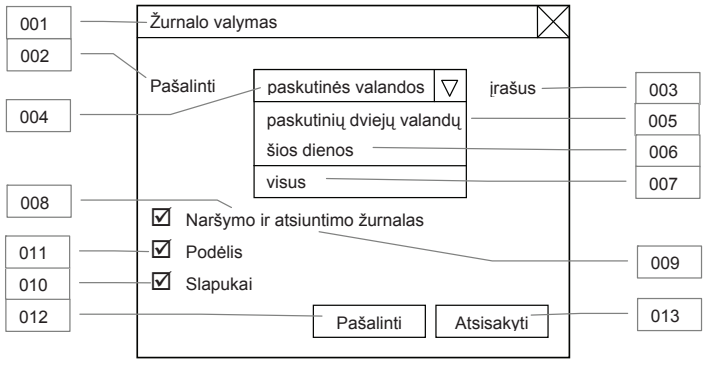
Programinės įrangos lokalizavimo principą pailiustruosime paprastu naršyklės fragmento pavyzdžiu (21 pav.). Pateiktas pavyzdys apima tik mažą dalį visų lokalizavimo aspektų, tačiau iliustruoja kai kurias problemas, su kuriomis nuolat susiduriama lokalizavimo metu. Paveikslo viršuje schematiškai pavaizduotas lokalizuojamos programos langas, apačioje – tame lange pavartotų tekstų (eilučių) failas, vadinamasis tekstinių lokalizuojamųjų išteklių failas. Kiekviena eilutė turi vardą (šiame pavyzdyje 001–013) ir tekstą, skirtą rodyti ekrane (tarp stačiųjų kabučių). Etiketės su eilučių vardais iš išteklių failo rodo, kurioje dialogo lango vietoje atitinkama eilutė pasirodo programos veikimo metu.



Išteklių failas (en-US)

```
001 "Clear History"
002 "Time range to clear: "
003 " "
004 "Last Hour"
005 "Last Two Hours"
006 "Today"
007 "Everything"
008 "Browsing &amp;amp; Download History"
009 "Browsing History"
010 "Cookies"
011 "Cache"
012 "Clear"
013 "Cancel"
```

21 pav. Programos grafinės sąsajos fragmento ir jį atitinkančio išteklių failo pavyzdys



Išteklių failas (lt-LT)

```
001 "Žurnalo valymas"
002 "Pašalinti"
003 "įrašus"
004 "paskutinės valandos"
005 "paskutinių dviejų  valandų"
006 "šios dienos"
007 "visus"
008 "Naršymo ir atsiuntimo žurnalas"
009 "Naršymo žurnalas"
010 "Slapukai"
011 "Podėlis"
012 "Pašalinti"
013 "Atsisakyti"
```

22 pav. Lokalizuotos programos grafinės sąsajos fragmento ir jį atitinkančio išteklių failo pavyzdys

Išteklių failai pateikiami atskirai. Lokalizuotojas paprastai dirba su daug išteklių failų ir eilučių ir nežino, kur kuri eilutė pasirodys programos grafinėje sąsajoje. Kaip eilutę išversti tenka spręsti vien iš eilutės teksto. Ne visada būna galimybė greitai pamatyti išverstas eilutes veikiančioje programoje, nes dauguma programų lokalizuojamos dar nebaigus programuoti originalo, kai dar ne visos programos funkcijos realizuotos, o numatomas funkcijas įvardijantys tekstai jau iškelti į lokalizuojamuosius išteklius.

22 paveiksle pavaizduotas to paties lango sulietuvintas variantas (viršuje) ir išteklių failas (apačioje).

Palyginus anglišką ir lietuvišką išteklių failus, galima pastebėti, kad vartojamos skirtingos frazių formos, pvz., *Today* = *šios dienos*, skirtingai vartojamos didžiosios ir mažosios raidės, keičiasi net tekstų semantika (plg. 21 pav. išteklių failo 002–003 eilutes ir 2 pav. išteklių failo 002–003 eilutes). Be to, dažnai pasitaiko tekstų alternatyvų, kurios skirtos rodyti ekrane priklausomai nuo konteksto (pvz., 008 ir 009 eilutės rodomos tame pačiame valdiklyje, tačiau kuri bus rodoma priklauso nuo programos naudojimo sąlygų: jei atsiuntimų žurnalas nėra tuščias, tai būtų rodoma 008 eilutė, priešingu atveju – 009).

Nežinant konteksto tinkamas lietuvių kalbos frazių formas parinkti sudėtinga, dažniausiai reikalingas daugkartinis testavimas ir koregavimas. Darbą apsunkina ir tai, kad ne visas eilutes lengva pastebėti programos testavimo metu, nes dalis tekstų pasirodo tik retais atvejais, esant tam tikroms, sunkiai modeliuojamoms situacijoms.

5.2. Lokalizuojamųjų išteklių atskyrimo metodai ir pateikimo formatai

Šiame skyrelyje vartojamos sąvokos *lokalizuojamųjų išteklių atskyrimo metodas* ir *lokalizuojamųjų išteklių pateikimo formatas*. Išteklių atskyrimo metodas teikia veiksmus ir priemones lokalizuojamiems ištekliams atskirti ir pateikti (įskaitant ir išteklių pateikimo formatą). Formatas yra daugiareikšmė sąvoka. Čia išteklių pateikimo formatu laikysime lokalizuojamuose ištekliuose laikomų duomenų apipavidalinimo būdą. Šis būdas gali priklausyti nuo išteklių atskyrimo metodo (t. y. formatą apibrėžia išteklių atskyrimo metodas).

Formatai įvairūs, priklauso nuo programavimo kalbos, kuria parašyta programa, naudojamo kompiliatoriaus, platformos, kuriai kuriama programinė įranga, išteklių atskyrimo metodo.

Prieš nagrinėjant lokalizuojamųjų išteklių formatus naudinga apžvelgti pagrindinius programinės įrangos internacionalizacijos tipus.

R. Laucius (2007), remdamasis D. Taylor (1992), skiria tris internacionalizacijos tipus:

1. Internacionalizacija kompiliavimo metu.
2. Internacionalizacija saistymo metu.
3. Internacionalizacija vykdymo metu.

Iš jų pastarieji du (2 ir 3) pasirodžius Tayloro knygai (1992 m.) ir kintant programinei įrangai bei kompiliatoriams susiliejo į vieną.

Internacionalizacija kompiliavimo metu – tai programos projektavimas, kai lokalizavimui skirti ištekliai neatskirti nuo pirminio programos teksto: teksto eilutės, kurios bus matomos kompiuterio ekrane, įkompiliuojamos į pirminį programos tekstą. Tuomet lokalizuojant programą daromos atskiros kopijos kiekvienai lokalei, pirminiame tekste randamos lokalizuotinos eilutės, jos lokalizuojamos (verčiamos) ir programa perkompiliuojama. Lokalizuoti programą, neperkompiliuojant jos, neįmanoma. Lokalizavimo metu atsiranda pavojus pažeisti pirminį programos tekstą.

Tokį internacionalizacijos tipą vadinti internacionalizacija yra daugiau nei simboliška, kadangi šiuo metu programą, kurioje tekstiniai ir kiti nuo kalbos ir kultūros priklausantys ištekliai nėra atskirti nuo pirminio teksto, negalima vadinti internacionalizuota. Tačiau tenka pastebėti, kad net ir dabar pasitaiko programų, kuriose lokalizuotinos eilutės tebėra paliktos pirminiuose programos tekstuose (Laucius, Dagienė, 2003). Taip pat lokalizavimo patirtis (Dagienė, Jevsikova, 2005; Jevsikova, 2003 ir kt.) rodo, kad net jeigu lokalizuojamieji ištekliai atskiriami nuo pirminių tekstų, kai kuriuos tekstus pamirštama atskirti, ir dėl to norint juos lokalizuoti tenka modifikuoti pirminį programos tekstą. Pagrindinė priežastis, dėl kurios net dabar pasirenkamas toks programos projektavimas, tai skubotas pirmos versijos išleidimas nesirūpinant išteklių atskyrimu ir projektavimu tarptautinei rinkai. Tačiau vėliau, kai atsiranda lokalizavimo poreikis, autoriai bando iškelti lokalizuotinus išteklius, o dėl nemetodiško pradinio projektavimo dalis jų lieka pirminiuose tekstuose.

Internacionalizacija saistymo ir vykdymo metu pasižymi tuo, kad lokalizavimui skirti ištekliai atskirti nuo pirminio programos teksto. Norint įtraukti lokalizuotus išteklius, pirminio programos teksto nereikia perkompiliuoti. Lokalizavimui

skirti ištekliai įkompiliuoti į vykdomąsias programas arba į vykdymo metu prie programų prijungiamas bibliotekas, išteklių paketus ar duomenų bazes.

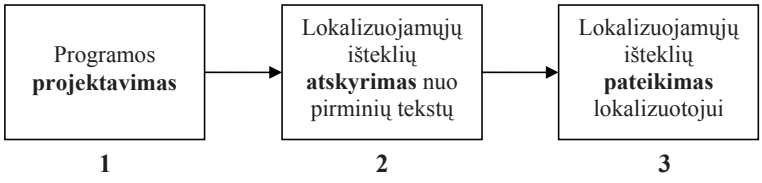
Daugelis stambesnių programinės įrangos gamintojų kuria savitus išteklių atskyrimo metodus ir išteklių pateikimo lokalizavimui formatus. Atvirųjų programų kūrėjai taip pat kuria ir naudoja savo formatus. Tačiau šiuo metu egzistuojantys formatai yra panašūs tuo, kad jie pateikia lokalizuotinus išteklius be konteksto arba tik su menkomis užuominomis apie kontekstą, kuriame tam tikras pranešimas bus naudojamas programoje.

23 paveiksle pavaizduotas programinės įrangos parengimo lokalizavimui procesas: projektavimo metu žinomas visų eilučių ir kitų nuo lokalės priklausomų elementų kontekstas programoje, tada lokalizuojamieji ištekliai atskiriami nuo pirminio programos teksto ir tam tikru formatu (kuriuos apžvelgsime toliau) pateikiami lokalizuotojui. Tarp 2 ir 3 žingsnio prarandamas lokalizuojamųjų išteklių elementų kontekstas. Taip pat atskirti ištekliai lokalizavimui gali būti pateikiami išteklių atskyrimo metodo numatomu formatu arba transformavus į kitą formatą (pvz., grynojo teksto). Pastaruoju atveju prarandama dar daugiau kontekstinės informacijos, kuri galėtų būti naudinga lokalizuotojui.

Autoriai (Ågerfalk ir kt., 2008) taip pat tvirtina, kad pagrindinė problema, su kuria susiduriama projektuojant tarptautinei rinkai skirtą programinę įrangą, – tai informacijos perdavimas tarp projektavimo etapų ir darbuotojų komandų.

Toliau apžvelgsime pagrindinius lokalizuojamųjų išteklių atskyrimo metodus ir formatus. Vienas metodas gali turėti keletą jam būdingų lokalizuojamųjų išteklių pateikimo failų formatų. Tačiau projektuotojai gali naudoti ir kitus formatus, pavyzdžiui, paprastesnį tekstinį formatą. Todėl iš programos lokalizuojamųjų išteklių pateikimo formato ne visada galima vienareikšmiškai nustatyti išteklių atskyrimo metodą.

Pateiksime lentelę, kurioje išvardyti pagrindiniai lokalizuojamųjų išteklių atskyrimo metodai ir atitinkami metodo numatytieji lokalizuojamųjų išteklių pateikimo failų formatai, kurie analizuojami tolesniuose skyreliuose (28 lentelė).



23 pav. Programinės įrangos parengimo lokalizavimui etapai

28 lentelė. Lokalizuojamųjų išteklių atskyrimo metodai ir atitinkami failų formatai

Išteklių atskyrimo metodas	**Pagrindiniai numatytieji išteklių failų formatai**
RC	RC, RESOURCES, EXE, DLL
RESX	RESX, RESOURCES, EXE, DLL
GNU „Gettext“	PO, POT, MO
Javos ištekliai	PROPERTIES
„Mozillos“ ištekliai	DTD, PROPERTIES
PHP	PHP
XLIFF	XLIFF

Lentelėje paminėtas XLIFF nėra priskiriamas prie išteklių atskyrimo metodų, jo pavadinime vartojamas žodis „formatas“ (angl. *XML Localization Interchange File Format*), nes nenumato priemonių ištekliams atskirti, o nurodo taisykles ištekliams pateikti. Papildžius XLIFF atitinkamomis išteklių atskyrimo priemonėmis, jį galima būtų laikyti ir išteklių atskyrimo metodu.

RC

Lokalizuojamųjų išteklių atskyrimo metodas RC sukurtas „Microsoft“ bendrovės ir paprastai naudojamas jos pačios programinėje įrangoje bei programinėje įrangoje, skirtoje „MS Windows“ operacinėms sistemoms. Metodo pavadinimas kilęs iš lokalizuojamųjų išteklių scenarijaus failo prievardžio RC. Tai tekstinis failas, kuriame gali būti naudojamos vienbaitė, dvibaitė ar keturbaitė koduotės[1]. Scenarijus apibrėžia teksto eilutes ir išorinius lokalizuojamuosius išteklius (piktogramas ir kitus grafinius išteklius, garso ir vaizdo išteklius, dialogo langus, šriftus ir kt. – iš viso 15 numatytųjų išteklių tipų). RC scenarijus turi keletą sekcijų, iš jų tris pagrindines sekcijas: 1) programos meniu, 2) statinių dialogo langų ir 3) teksto eilučių (pranešimų).

Išteklių scenarijai RC kompiliuojami ir ištekliai susiejami su vykdomaisiais failais ar dinaminėmis bibliotekomis (DLL). Tada lokalizuojamieji ištekliai pateikiami kaip atskiros dinaminių bibliotekų arba vykdomųjų failų sekcijos (išteklių sekcijos). Išteklius galima lokalizuoti arba iš pradinių tekstinių failų, arba iš dvejetainių failų pasitelkus specializuotas išteklių tvarkytuves – tokių išteklių modifikavimo vizualias priemones, pavyzdžiui, „Resource Hacker“, „Resource Tuner“ ir kt.

[1] Microsoft Corporation. Go Global Development Center. http://msdn.microsoft.com/en-us/goglobal/bb688117.aspx [žiūrėta 2010-06-29].

Išteklių scenarijaus meniu sekcijoje (žr. pavyzdį 24 paveiksle) kiekvienas meniu elementas turi jį atitinkančią originalo eilutę, pvz., „&Save“. Jeigu meniu elementas gali būti pasiekiamas prieigos klavišu, tai eilutėje prieš klavišą įvardijantį ženklą (raidę) rašomas ampersendo ženklas &. Atskirai nurodomi komandų klavišai, pvz., „Ctrl+S“.

Dialogo lango sekcija (25 pav.) turi valdiklio pavadinimą, valdiklio koordinates ir dydžio parametrus bei tekstą, vaizduojamą tame valdiklyje. Yra numatyta 19 valdiklių tipų: 11 pagrindinių valdiklių tipų (mygtukas, išskleidžiamasis sąrašas, žymimasis langelis ir kt.) bei jų variacijos (centruotas teksto užrašas, lygiuotas pagal kairįjį kraštą teksto laukas, trijų būsenų valdiklis – 3 žymimųjų akučių rinkinys ir kt.).

Eilučių sekcijoje pateikiamos eilutės, kurios paprastai nėra statiškai priskirtos prie atitinkamų statinių dialogo langų valdiklių (26 pav.). Į šią sekciją patenka programos pranešimai apie klaidas (176–182 eilutės pavyzdžio paveiksle), klausimai programos naudotojui, įvairių veiksenų pavadinimai (183–188), sąrašų elementai (189–191), parametrizuotos (179–182) ir dinamiškai atsirandančios dialogo languose eilutės (186–188).

```
6 MENU
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
{
POPUP "&File"
{
    MENUITEM "&New...\tCtrl+N", 57600
    MENUITEM "&Open...\tCtrl+O", 57601
    MENUITEM "&Save\tCtrl+S", 57603
    MENUITEM "Save &As...", 57604
    MENUITEM SEPARATOR
    MENUITEM "&Print...\tCtrl+P", 57607
    MENUITEM "Print Pre&view", 57609
    MENUITEM "Page Set&up...", 32771
    MENUITEM SEPARATOR
    MENUITEM "Recent File", 57616, GRAYED
    MENUITEM SEPARATOR
    MENUITEM "Sen&d...", 57612
    MENUITEM SEPARATOR
    MENUITEM "E&xit", 57665
}
MENUITEM SEPARATOR
}
```

24 pav. „MS Windows“ RC išteklių failo meniu sekcijos pavyzdys

```
143 DIALOG 0, 0, 185, 112
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUPWINDOW | WS_VISIBLE |
WS_DLGFRAME
CAPTION "Paragraph"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "MS Shell Dlg"
{
  GROUPBOX   "Indentation", 1022, 7, 7, 114, 75
  LTEXT   "&Left:", -1, 13, 21, 29, 8
  EDITTEXT   1000, 52, 19, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&Right:", -1, 13, 42, 29, 8, NOT WS_GROUP
  EDITTEXT   1001, 52, 40, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&First line:", -1, 13, 63, 32, 8, NOT WS_GROUP
  EDITTEXT   1002, 52, 60, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&Alignment:", 1017, 13, 91, 37, 8
  COMBOBOX   111, 52, 89, 60, 45, CBS_DROPDOWN | WS_VSCROLL
  DEFPUSHBUTTON   "OK", 1, 128, 10, 50, 14
  PUSHBUTTON   "Cancel", 2, 128, 27, 50, 14
}
```

25 pav. „MS Windows“ RC išteklių failo dialogo lango sekcijos pavyzdys

```
STRINGTABLE
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
{
176, "There are too many tab stops set in this paragraph."
177, "An error occurred while sending the document."
178, "There is not enough memory. Quit one or more programs and then try again."
179, "Can not load %1 files."
180, "Unable to open %1.  There are too many files already open."
181, "Unable to create %1. This folder is full. Use another folder or delete some files."
182, "The document %1 is in use by another application and cannot be accessed."
183, "Text"
184, "Rich Text"
185, "Word"
186, "Options"
187, "Write"
188, "Embedded"
189, "Text Document"
190, "Rich Text Document"
191, "Word 6 Document"
}
```

26 pav. „MS Windows“ RC išteklių failo teksto eilučių sekcijos pavyzdys

Eilučių sekcija yra pati sudėtingiausia lokalizavimo požiūriu (Esselink, 2000, p. 63), kadangi joje nebėra grafinės sąsajos konteksto, kuris iš dalies buvo pateikiamas RC dialogo langų ir meniu sekcijose.

Pranešimams dažnai trūksta aiškumo, norint juos kokybiškai lokalizuoti reikia nemažai programavimo žinių, informacijos apie konkrečią programą ir pranešimą su-

pantį kontekstą joje. Priklausomai nuo konteksto dažnai skirtingai verčiami trumpi (vieno ar dviejų žodžių) pranešimai. Pavyzdžiui, pranešimas „None“ gali turėti kelias prasmes ir į kitas kalbas gali būti verčiamas skirtingai (nieko, joks, nėra). Žodis „copy“ gali būti verčiamas veiksmažodžiu (kopijuoti), daiktavardžiu (kopija) arba sudaryti dalį kurios nors kitos frazės, skirtos rodyti ekrane (atšaukti kopijavimą). Siekiant išvengti klaidų tenka nemažai laiko skirti lokalizuotos programos testavimui.

Semantinis eilučių kontekstas nėra pateikiamas nei vienoje iš sekcijų. Tai metodo trūkumas. Jo privalumas, palyginus su kitais metodais – numatyta galimybė atskirti ne tik tekstinius išteklius, bet ir dvejetainius, lokalizavimo metu vaizdžiai pateikti statinius dialogo langus pasinaudojus egzistuojančiomis priemonėmis.

RESX

Tai RC metodo atmaina, kai išteklių apibrėžimo scenarijui aprašyti naudojama XML kalba (Hall, 2007). Formatas buvo sukurtas „.NET“ platformos programoms, bet gali būti naudojamas ir kitose programose.

Ištekliams pateikti naudojamas RESX failų formatas, kuriame objektai ir eilutės ženklinamos XML gairėmis. Į RESX failą gali būti įterpta ir dvejetainių objektų (paveikslų, garsų, piktogramų ir pan.). Išteklių failą galima taisyti įprasta tekstų rengykle arba specialiomis priemonėmis, vizualizuojančiomis ir dvejetainius objektus.

Lokalizuojamieji ištekliai pateikiami vardų ir jų reikšmių poromis (27 pav.).

Tekstinėms eilutėms nurodomas vardas ir eilutė, dvejetainiams objektams – vardas ir MIME tipas bei pateikiamas pats objektas įterptuoju dvejetainiu kodu.

Lokalizuotojui pateikiami RESX formato ištekliai gali būti konvertuojami į tekstinį failą, lentelę arba pateikiami tiesiogiai. Pastaruoju atveju su jais dirbama naudojant specializuotas tokių išteklių tvarkytuves. RESX failas taip pat gali būti kompiliuojamas į dvejetainį RESOURCES tipo failą, įkompiliuojamas į vykdomuosius failus (EXE) arba dinamines bibliotekas (DLL).

```
< ?xml version="1.0" encoding="utf-8" ? >
<root>
   <xsd:schema > …. </xsd:schema>
   …
  <data name="download">
    <value>Drop a link or file to download it</value>
  </data>
…
</root>
```

27 pav. RESX formato išteklių fragmento pavyzdys

GNU „Gettext“

GNU „Gettext“ metodas ir kartu priemonių paketas automatiškai atskiria teksto eilutes, skirtas rodyti ekrane, nuo pirminio programos teksto. Paplitęs atvirųjų programų lokalizuojamiems tekstiniams ištekliams pateikti. Yra sukurta „Gettext“ priemonių paketų ir bibliotekų, pateikiančių sąsajas įvairioms programavimo kalboms (Javai, C, C++, Paskaliui, PHP, Phytonui, Perlui ir kt.).

GNU „Gettext“ priemonių paketą sudaro (Drepper ir kt., 2010):

1. Taisyklių rinkinys, kaip rašyti programas ir atskirti tekstinius išteklius į „Gettext“ katalogus.
2. Eilučių katalogų failų ir aplankų struktūros bei jų vardų sudarymo taisyklės.
3. Biblioteka kreipiniams vykdymo metu į išverstas eilutes.
4. Keletas atskirų programinių priemonių eilučių vertimams tvarkyti.
5. Biblioteka, skirta išverstų eilučių failų kūrimui tvarkyti.
6. „Emacs“ grupės tekstų rengyklės, skirtos eilutėms atnaujinti.

Eilučių sąrašas gaunamas automatiškai iš programos tekste panaudotų eilučių, ištraukiant jas į POT failus panaudojus funkciją „Gettext“ arba kitas funkcijas, kurių vardai prasideda pabraukimo brūkšniu, pvz., „_Eilutės tekstas“. Šios funkcijos atlieka ir atvirkščią veiksmą – randa reikiamas eilutes, kai jas reikia parodyti veikiančios programos ekrane.

Lokalizuotojas dirba su PO formato failais, kurie yra sukuriami iš POT šablono ir yra jam identiški pagal formatą. POT šablonas yra pagrindinė eilučių saugykla, kurioje laikomos tik originalo eilutės (be jų vertimų), o PO failai – tai šio šablono kopijos, su kuriomis dirba įvairių kalbų lokalizuotojai ir kurios papildomos eilučių vertimais. Kai reikia atnaujinti lokalizuojamųjų eilučių failą, atnaujinamas POT šablonas, o iš jo naujų eilučių, parašytų originalo kalba, tekstas importuojamas į PO failus.

PO formato failas – tai tekstinis failas, kuriame kiekvieną įrašą atitinka keletas failo teksto eilučių – laukų. Vienoje eilutėje (msgid) rašomas tekstas originalo kalba, kitoje (msgstr) – teksto vertimas (28 pav.).

Papildomi laukai žymimi specialiaisiais simboliais. Pavyzdžiui, eilutėje, prasidedančioje ženklu „#“, rašomi vertėjo komentarai, po ženklų poros „#.“ rašomi automatiniai komentarai, kuriuos sukuria PO failą tvarkanti programa, po ženklų poros „#,“ rašomos įrašo gairelės, pavyzdžiui, gairelė „fuzzy“ reiškia tikslintiną eilutę. Specialios programos (pvz., „Poedit“) gali interpretuoti PO failo įrašus ir vaizdžiai juos pateikti lentele. Jos turi priemonių įrašams ieškoti, tvarkyti ir kt.

Iš PO tekstinių failų yra kompiliuojami dvejetainiai MO tipo failai, kurie skirti platinti su programa. Taip gaunama failų seka POT → PO → MO.

```
#: ./skins/plone_scripts/folder_rename.cpy:58
msgid "${count} item(s) renamed."
msgstr "Pervardyta elementų: ${count}."

#: ./skins/plone_scripts/folder_delete.cpy:49
msgid "${items} could not be deleted."
msgstr "${items} negalima pašalinti."

#. Default: "${monthname} ${year}"
#: ./skins/plone_portlets/portlet_calendar.pt:56
msgid "${monthname} ${year}"
msgstr "${year} ${monthname}"

#: ./skins/plone_scripts/check_id.py:62
msgid "${name} is not a legal name. The following characters are
invalid: ${characters}"
msgstr "„${name}“ varde yra neleistinų ženklų: ${characters}"

#: ./skins/plone_scripts/check_id.py:70
msgid "${name} is reserved."
msgstr "„${name}“ yra rezervuotas vardas."
```

28 pav. „Gettext“ PO formato pavyzdys (turinio valdymo sistemos „Plone“ eilučių failo fragmentas)

Tai, kad PO faile originalo kalba užrašyta eilutė (msgid) atlieka ir viso įrašo vardo funkciją, palengvina darbą programuotojams, tačiau lokalizuojant iš anglų kalbos su „Gettext“ iškyla problema. Ta pati anglų kalbos eilutė, naudojama skirtingose programos vietose, dažnai būna verčiama skirtingai, pvz., „Open file“ – „Atverti failą“ (meniu komanda), „Failo atvėrimas“ (lango pavadinimas). Norint įdėti keletą vienodų eilučių, reikia papildomų programavimo pastangų.

Kitas „Gettext“ metodo trūkumas yra tas, kad jis numato tik tekstinių lokalizuojamųjų išteklių atskyrimą. Kiti nuo lokalės priklausomi ištekliai (paveikslai, piktogramos, garsai ir kt.) turi būti atskiriami kitais metodais, kuriuos pasirenka programuotojas. Taip pat šis metodas nenumato lokalės svarbiausių elementų (datos ir laiko formatų, piniginių vienetų ir kt., žr. 4.2 skyr.), todėl dažniausiai kartu su juo naudojamas POSIX lokalės modelis.

Metodo privalumas – juo galima pateikti skirtingas formas žodžių, vartojamų su skaičiais (pvz., 1 objektas, 2 objektai, 10 objektų). Kalbos formų kaitymas aprašomas scenarijumi, kuris lietuvių kalbai būtų toks:

```
Plural-Forms: nplurals=3; plural=n%10==1 && n%100!=11
? 0 : n%10>=2 && (n%100<10 || n%100>=20) ? 1 : 2.
```

Jį įterpus į PO failo antraštės sekciją toliau galima nurodyti visas tris bet kurio žodžio formas.

Javos ištekliai

Programuojant Java sistema naudojamas lokalizuojamų išteklių atskyrimo metodas, kurio pagrindinės savybės:

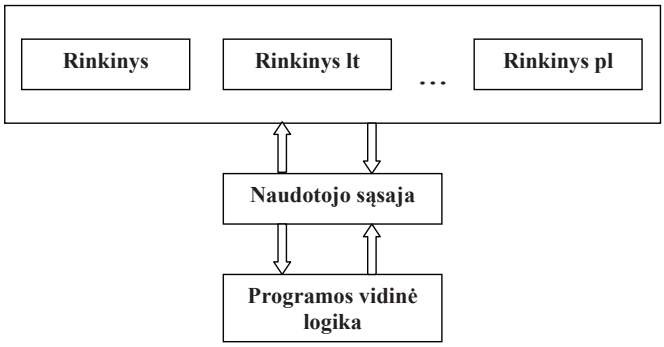
1. Yra priemonės lokalės informacijai laikyti ir užklausti;
2. Programoje galima naudoti daugelį lokalių;
3. Galima pridėti naujų lokalių.

Javos internacionalizuotos programos išteklių struktūra pavaizduota 29 paveiksle.

Ištekliams atskirti naudojama Javos abstrakčioji klasė *ResourceBundle*. Ištekliai gali būti pateikiami dvejopai: Javos klasėmis arba tekstiniais failais su prievardžiu PROPERTIES. Tekstinių išteklių failų formatas – eilučių vardų ir eilučių tekstų porų sąrašas (30 pav.). Papildomai toks failas gali turėti komentarų. Šiuo metu tokie failai paprastai koduojami UTF-8 koduote, tačiau anksčiau buvo naudojamas ženklų, kurių nėra ASCII koduotėje, keitimas unikodo pakaitos sekomis *\uFFFF*, pvz., vietoje raidės *ė* būtų rašoma *\u0117*. Programų, kuriose tebenaudojamas toks kodavimas, dar pasitaiko ir dabar.

Lokalizuotiems ištekliams įkelti programos vykdymo metu naudojami tam skirti specialieji Javos klasės metodai. Esant reikalui, klasę *ResourceBundle* galima praplėsti pritaikant konkretiems išteklių atskyrimo poreikiams.

Javos išteklių rinkinių metodas pastaraisiais metais taikomas ne tik programuojant Java. Panašus principas gali būti naudojamas ir programuojant C bei C++ kalbomis, taip pat yra panašumų „Mozilla“ išteklių atskyrimo metode.



29 pav. Javos programos lokalizuojamųjų išteklių struktūra

```
nv_done=Done
nv_timeout=Timed Out
nv_stopped=Stopped
openFile=Open File

droponbookmarksbutton=Drop a link to bookmark it
dropondownloadsbutton=Drop a link or file to download it
droponnewtabbutton=Drop a link or file to open it in a new tab
droponnewwindowbutton=Drop a link or file to open it in a new window
droponhomebutton=Drop a link or file to make it your home page
droponhometitle=Set Home Page
droponhomemsg=Do you want this document to be your new home page?
```

30 pav. Javos PROPERTIES failo formato pavyzdys

„Mozillos“ ištekliai

Visose „Mozillos“ bendrijos kuriamose atvirosiose programose (naršyklėje, elektroninio pašto programose, hiperteksto rengyklėje, pokalbių programoje, kalendoriuje, nedideliuose papildiniuose) naudojamas tas pats išteklių atskyrimo metodas ir pateikimo formatas. Šį metodą pateiksime išsamiau, nes „Mozilla“ programų, jų lokalizacijų ir naudotojų skaičius nuolat didėja.

Tekstiniai lokalizuojamieji ištekliai iškeliami į dviejų pagrindinių tipų grynojo teksto failus DTD ir PROPERTIES.

DTD failuose pateikiamos statinės išteklių eilutės, t. y. tokios, kurios panaudotos grafinės sąsajos aprašo faile: pastovios meniu komandos, dialogo langų elementų (pvz., mygtukų, teksto užrašų, žymimųjų langelių) pavadinimai. DTD failas koduojamas UTF-8 koduote ir turi daugeliui lokalizuojamųjų išteklių atskyrimo metodų būdingą eilučių vardų ir tekstų porų struktūrą, tik apsuptą XML gairėmis (31 pav.).

PROPERTIES failai nesiskiria nuo anksčiau aprašyto Javos išteklių rinkinių PROPERTIES failo formato. Tik „Mozillos“ programose laikomasi susitarimo, jog PROPERTIES failuose pateikiamos dinaminės naudotojo sąsajos eilutės, kuriomis operuoja „JavaScript“ scenarijai: programos pranešimai, pranešimai apie klaidas, sąrašų elementai, besikeičiantys valdiklių pavadinimai. Failas koduojamas UTF-8 koduote (anksčiau buvo koduojama unikodo kaitos ženklų sekomis).

„Mozillos“ metodas numato ne tik tekstinių išteklių atskyrimą, bet ir dvejetainių objektų, grafinio apipavidalinimo stilių, nuostatų failų, RDF failų ir pan. atskyrimą. Visi šie nuo lokalės priklausomi ištekliai skirstomi į aplankus ir pakuojami į nepriklausomus nuo platformos XPI paketus, skirtus platinti.

```
<!ENTITY  colorsDialog.title          "Spalvos">
<!ENTITY  window.width                "42em">
<!ENTITY  window.macWidth             "45em">

<!ENTITY  allowPagesToUse.label       "Spalvas rodyti iš tinklalapių">
<!ENTITY  allowPagesToUse.accesskey   "S">

<!ENTITY  color                       "Tekstas ir fonas">
<!ENTITY  textColor.label             "Tekstas:">
<!ENTITY  textColor.accesskey         "T">
<!ENTITY  backgroundColor.label       "Fonas:">
<!ENTITY  backgroundColor.accesskey   "F">
<!ENTITY  useSystemColors.label       "Spalvas imti iš operacinės sistemos">
<!ENTITY  useSystemColors.accesskey   "o">

<!ENTITY  underlineLinks.label        "Saitus pabraukti">
<!ENTITY  underlineLinks.accesskey    "b">
<!ENTITY  links                       "Saitų spalvos">
<!ENTITY  linkColor.label             "nelankytų:">
<!ENTITY  linkColor.accesskey         "n">
<!ENTITY  visitedLinkColor.label      "aplankytų:">
<!ENTITY  visitedLinkColor.accesskey  "a">
```

31 pav. DTD failo fragmentas („Mozilla Firefox“ naršyklės lietuviška lokalizacija)

XLIFF

Tai XML pagrindu sukurtas failų formatas lokalizavimo duomenų mainams tarp programinės įrangos kūrėjų ir lokalizuotojų arba tarp įvairių lokalizavimo automatizavimo programinių priemonių. Juo naudojantis galima lokalizuojamuosius išteklius, o taip pat ir bet kurį rišlų tekstą (pvz., interneto svetainių, dokumentacijos, įprastų tekstinių dokumentų) versti bei redaguoti nepriklausomai nuo jų formatų, naudojamų konkrečioje programoje. Keičiantis informacija šis formatas atlieka tarpininko vaidmenį tarp įvairių formatų.

Kuriant XLIFF standartą buvo turimas omenyje pramoninis lokalizavimo procesas, kai lokalizavimą atlieka specializuotos žmonių grupės ir vertėjas nebūtinai turi būti programuotojas, kai lokalizavimas pavedamas kelioms skirtingoms kompanijoms, kad tas pats formatas apimtų ir rišlaus teksto dokumentų vertimus, kad būtų galima patogiai ir operatyviai keistis duomenimis apie lokalizavimo eigą ir užbaigtumą.

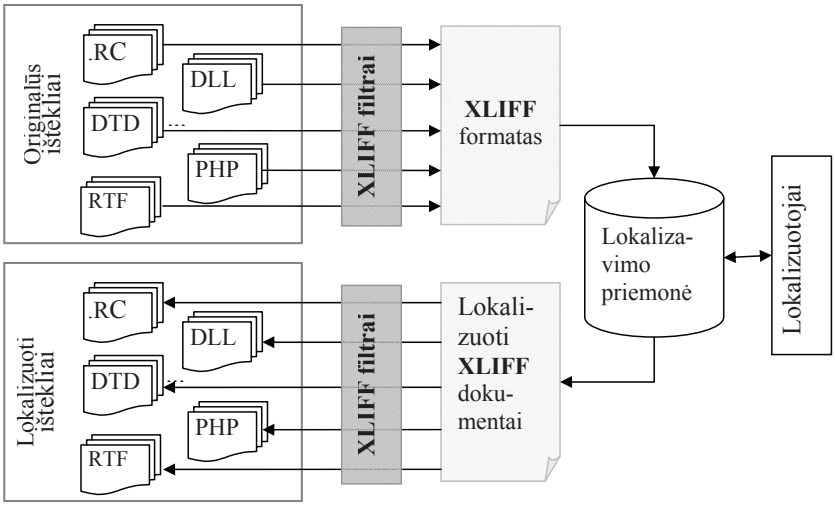
Šiuo metu naudojamas tik kaip tarpinis formatas: lokalizuojamieji ištekliai atskiriami naudojant vieną iš anksčiau aprašytų metodų ir pateikiami tą metodą atitinkančiais formatais (pvz., PO, RC, RESX, EXE, DLL, PROPERTIES, DTD), po to konvertuojami į XLIFF formatą naudojant specialias priemones – XLIFF filtrus, tada XLIFF failai pateikiami lokalizuotojams, kurie lokalizuoja (verčia) šį failą pasitelkę specializuotas lokalizavimo priemones, o lokalizavus XLIFF ištekliai konvertuojami atgal į originalų formatą naudojant atgalinius filtrus (32 pav.).

XLIFF specifikacijoje daug dėmesio skiriama verstino ar lokalizuotino failo formatavimo informacijai, kuri gali būti pateikiama HTML ar XML gairėmis, RTF ar WORD formatavimo direktyvomis ir pan. Tai labiau tinka rišlių tekstų vertimų darbams (gali būti taikoma elektroniniams žinynams lokalizuoti).

Kai pradinių lokalizuojamųjų išteklių failas konvertuojamas į XLIFF formatą, jo originalūs struktūriniai formatai (pvz., HTML gairės `<body>`) įrašomi į atskirą karkaso (angl. *skeleton*) failą. Šiame faile nurodoma, kur vertimo vienetas (angl. *unit*) turės būti įdėtas, konvertuojant lokalizuotą XLIFF failą atgal į originalų formatą. Struktūrinio formatavimo elementai (gairės) nėra pasiekiami lokalizuotojui (vertėjui) ir lokalizavimo priemonių kūrėjui. Jie dirba tik su tekstu ir XLIFF apibrėžtais elementais. Tokiu būdu apsisaugoma nuo dokumento pirminio teksto struktūros ir formatavimo informacijos pažeidimo.

Įterptinis formatavimas valdomas dviem būdais:

1. Įtraukiant į karkaso failą, t. y. vietoj originalios formato gairės įterpiama viena iš dviejų XLIFF gairių (*g* gairė poriniams formatavimo elementams, pvz., elementams , o *x* gairė – neporiniams, pvz., <img>).
2. Paliekant formatavimą XLIFF faile ir įdedant XLIFF gaires formatavimo pradžiai ir pabaigai pažymėti (pvz., `<it id="a1" pos="1">&lt; img src="pav.gif"&gt;</it>`).



32 pav. XLIFF formato naudojimas lokalizavimo procese

XLIFF turi priemonių lokalizavimo procesui žymėti. Specialūs lokalizavimo etapų elementai turi atributus, skirtus informacijai apie lokalizavimo laiką, naudotą priemonę, atsakingą asmenį, etapo pavadinimą (pvz., vertimas, recenzavimas), viso lokalizavimo proceso pavadinimą. Pastabų elementai skirti lokalizavimo etapui pakomentuoti laisva forma. Kiekvieno elemento vertimas gali turėti lokalizavimo etapo pavadinimo atributą, rodantį, kurio etapo metu buvo atliktas vertimas (tai gali būti svarbu recenzavimui, peržiūrai).

Pagrindinis XLIFF failo elementas yra `<trans-unit>` (vertimo vienetas), kuris turi įdėtinius elementus `<source>` ir `<target>`. Failas yra dvikalbis: `<source>` elemente laikoma originali eilutė arba jos segmentas (kai eilutė yra ilga, pvz., rišlaus teksto dokumente), o `<target>` – eilutės arba jos segmento vertimas. Kiekvienam originaliam lokalizuotinam elementui galima priskirti keletą vertimo (lokalizavimo) variantų: apytikslis vertimas, atitikmuo iš vertimo atminties, atitikmuo iš vertimo automato ir pan., iš kurių vertėjas gali suformuoti galutinį to elemento.

Kontekstui nurodyti XLIFF turi elementą `<context>`, kuriuo galima apibūdinti pradinio teksto `<source>` arba alternatyvaus vertimo kontekstą. Šio elemento įvedimo tikslas – leisti tam tikroms teksto dalims turėti skirtingus vertimus, priklausomai nuo to, iš kur jie ateina, pvz., skirtingai išversti tą patį tekstą, kuris naudojamas saityno formoje, dialogo lange, duomenų bazės formoje ir pan. Konteksto elementu pažymėta informacija yra skirta vertėjui arba lokalizavimo priemonei, jei ji suprojektuota taip, kad galėtų ją interpretuoti ir pasiūlyti tinkamą vertimą.

Tos pačios rūšies lokalizuojamiems ištekliams grupuoti yra numatytas elementas `<group>`.

Prie teksto fragmento gali būti priskirtas atributas, nurodantis valdiklio tipą, kuriame bus pavartota eilutė (pvz., mygtukas, žymimoji akutė, meniu elementas ir pan.).

Į XLIFF failą galima sudėti ne tik tekstinius lokalizuojamuosius išteklius, yra numatytas ir dvejetainiams objektams žymėti skirtas elementas `<bin-uint>` (objektas įterpiamas tiesiogiai, panašiai, kaip RESX formate).

Konvertuojant lokalizuotą XLIFF failą atgal į originalų formatą, jis suliejamas su karkaso failu ir pateikiamas pageidaujamu formatu. Konvertavimo filtras skaito žymes ir įdeda visus vertimus, kurie turi patvirtintųjų statusą. Jei nėra patvirtinto vertimo, įdedamas originalus tekstas.

XLIFF formatas turi privalumų palyginus su anksčiau nagrinėtais formatais, kadangi tai universalus lokalizavimo išteklių mainų formatas, galintis laikyti daugiau išteklių ir lokalizavimo proceso metainformacijos. Tačiau šiuo metu naudojamas tik

kaip tarpinis lokalizavimo formatas, todėl realiai metainformacija beveik nenaudojama (nes konvertuojama automatiškai iš esamų formatų, kuriuose šios informacijos nėra). Darbui su XLIFF skirtos priemonės šiuo metu panaudoja tik nedaugelį jo elementų. Be to, formato universalumas yra sąlyginis, kadangi atsiranda tarpinės priemonės – filtrai, kurie konvertuoja išteklius iš esamų formatų. Daugumą lokalizuotų XLIFF formatu failų po atgalinio konvertavimo tenka taisyti (išdėstymas ir formatas gali šiek tiek skirtis nuo originalaus). Lengviausiai konvertuojami ir nereikalauja taisymų XML, HTML ir RTF formatai.

XLIFF oficiali specifikacija yra įteisinta kaip OASIS standartas.

Kiti lokalizuojamųjų išteklių pateikimo formatai

Yra ir daugiau lokalizuojamųjų išteklių formatų. Jie visi turi vardų ir reikšmių porų sąrašus. Skiriasi tik tuo, kad vardai ir reikšmės atskiriamos kitais simboliais. LNG ir LANG formatų (naršyklė „Opera“, failų ir aplankų tvarkymo programa „Total Commander“, pokalbių programa „Skype“ ir kt.), CSV formato failuose eilutės vardas nuo eilutės teksto skiriami kableliais, TAB formatuose eilutės vardas nuo eilutės teksto skiriami tabuliacijos ženklais.

Lokalizuojant internete veikiančias sistemas tenka pastebėti, kad daugelis jų naudoja PHP formatą (pvz., „Moodle“ virtualioji mokymosi aplinka, „MediaWiki“ vikio sistema ir kt.). Tai paprasčiausias tekstinis failas, į kurį įrašytas PHP eilučių masyvas, taip pat turintis vardų ir reikšmių porų struktūrą (33 pav.).

```
$string['activitysince'] = 'Activity since $a';
$string['add'] = 'Add';
$string['addactivity'] = 'Add an activity...';
$string['addadmin'] = 'Add admin';
$string['addcreator'] = 'Add course creator';
$string['added'] = 'Added $a';
$string['addedrecip'] = 'Added $a new recipient';
$string['addedrecips'] = 'Added $a new recipients';
$string['addedtogroup'] = 'Added to group $a';
$string['addedtogroupnot'] = 'Not added to group $a';
$string['addedtogroupnotenrolled'] = 'Not added to group $a,
because not enrolled in course';
$string['addinganew'] = 'Adding a new $a';
$string['addinganewto'] = 'Adding a new $a->what to $a->to';
$string['addingdatatoexisting'] = 'Adding data to existing';
$string['addnewcategory'] = 'Add new category';
$string['addnewcourse'] = 'Add a new course';
$string['addnewuser'] = 'Add a new user';
```

33 pav. PHP tekstinių lokalizuojamųjų išteklių pateikimo formatas

29 lentelė. Lokalizuojamųjų išteklių formatai dažniau naudojamose programose

Programinė įranga	Išteklių pateikimo formatas
Drupal	Gettext PO
Internet Explorer	RC, RESX
Lemill	Gettext PO
MediaWiki	PHP
Moodle	PHP
Mozilla Firefox	Mozilla DTD, PROPERTIES
Mozilla Thunderbird	Mozilla DTD, PROPERTIES
NVU	Mozilla DTD, PROPERTIES
Opera	LNG
Outlook Express	RC, RESX
Pidgin	Gettext PO
Plone	Gettext PO
SeaMonkey	Mozilla DTD, PROPERTIES
Skype	LANG

PHP funkcija *get_string* (*'eilutės vardas'*, *'failo vardas'*) naudojama atitinkamai teksto eilutei iš PHP failo gauti.

Reikia pastebėti, kad programos gali naudoti vieną lokalizuojamųjų išteklių atskyrimo metodą, tačiau lokalizuotojams tekstiniai ištekliai pateikiami supaprastintu tekstiniu formatu – tokiu būdu kartais susiaurinama ir šiaip negausi kontekstinė informacija.

29 lentelėje pateikiamas populiarių programų ir jų lokalizuojamųjų išteklių pateikimo formatų pavadinimai.

5.3. Kontekstinė informacija lokalizuojamuose ištekliuose

Programinės įrangos tekstai, pateikiami dialogo languose, yra lakoniški, atsieti nuo konteksto, juose gausu naujų terminų (kurių gali dar nebūti kalboje, kuriai lokalizuojama). Todėl programos lokalizuotojas susiduria su šiomis pagrindinėmis problemomis:

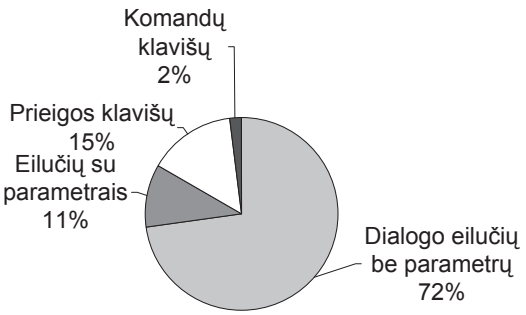
1. Mato tik atskirus žodžius ar frazes be konteksto – lokalizuodamas programą, žmogus dirba su dialogo eilučių duomenų baze.

2. Lokalizuojamų tekstų kontekstas pamatomas tik programai veikiant ir (iš dalies) nagrinėjant pirminius programos tekstus (jeigu tai leidžia daryti programos licencija), o tai labai sunkus ir imlus darbas.
3. Dalis programos dialogo tekstų atsiranda tik esant ypatingoms situacijoms (klaidoms, kitų programų poveikiui), kurias sudėtinga arba neįmanoma sumodeliuoti testuojant lokalizuotą programą.

Kadangi dauguma lokalizuojamų programų lokalizuojama iš anglų kalbos, tai dar viena grupė problemų, kurios atsiranda lokalizavimo metu, kyla iš anglų kalbos neapibrėžtumo ir jos kompiuterininkų (programuotojų) žargono. Anglų kalboje gausu sinonimų, tuo pačiu žodžiu vadinami skirtingi dalykai, dažnai, kai nėra konteksto, sudėtinga suprasti, ar žodis yra veiksmažodis, ar daiktavardis (angliškai vienodai rašoma, pvz., *File* (*failas* – daiktavardis, *įtraukti* [į aplanką] – veiksmažodis), *login – prisijungti* (veiksmažodis), *prisijungimas*, *prisijungimo vardas* (daiktavardis)), anglų kalbos leksika įvairuoja, net toje pačioje programoje tai pačiai funkcijai įvardyti gali būti vartojami skirtingi pavadinimai, kompiuterių terminijoje gausu metaforų, žargono, terminija stokoja sistemingumo (Grigas, 2008).

Lokalizavimo metu kontekstą sunku suvokti, kadangi eilutės yra trumpos. Apibūdinsime lokalizuojamųjų išteklių struktūrą remdamiesi programų pavyzdžiais: naršykle „Mozilla Firefox“ (kliento programa) ir virtualiąja mokymosi aplinka „Moodle“ (serverio programa).

Išanalizavę interneto naršyklės „Mozilla Firefox“ trijų iš eilės leistų versijų lokalizuojamųjų išteklių struktūrą, eilučių tipų procentinį pasiskirstymą pateikiame 34 paveiksle.



34 pav. Lokalizuojamųjų tekstinių išteklių eilučių pasiskirstymas remiantis „Mozilla Firefox“ pavyzdžiu

17 proc. visų išteklių eilučių sudaro prieigos ir komandų klavišai, kuriuos reikia parinkti lokalizavimo metu. Ir ne tik parinkti juos prasmingus, bet ir patikrinti, ar jie nesikartoja jų galiojimo srityje.

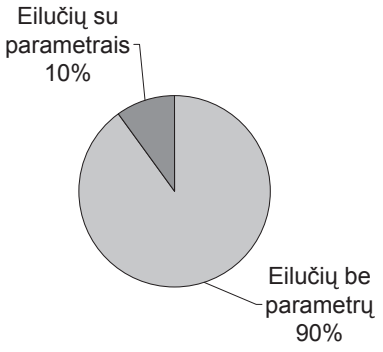
11 proc. eilučių viduje yra pavartotas vienas ar keli parametrai – kintamieji, kurių vietoje bus įrašyta arba kita eilutė iš išteklių, arba tam tikra reikšmė, kuri gali būti skaičiuojama programos vykdymo metu. Kiekvienoje tokioje eilutėje yra vidutiniškai 1,4 parametrai. Tai potencialių klaidų lokalizavimo metu eilutės, kadangi parametro reikšmė dažniausiai išaiškėja tik programai veikiant.

Kitos eilutės – tai eilutės be parametrų, atitinkančios programos dialogų tekstus. Lokalizavimo komentarų skaičius sudaro tik 4 proc. visų lokalizuojamųjų išteklių eilučių. Tai reiškia, kad visoje likusioje eilučių dalyje trūksta konteksto. Remiantis „Moodle“ virtualiosios mokymosi aplinkos lokalizuojamųjų išteklių tyrimais, gavome panašius rezultatus eilučių su parametrais atveju (35 pav.).

Lokalizavimo komentarų „Moodle“ lokalizuojamuose ištekliuose nėra, o eilučių vardai yra neprasmingi (arba atkartoja eilutę, arba vienas du žodžiai, nenusakantys eilutės konteksto programoje).

Kadangi „Moodle“ sistema yra serveryje veikianti programa, tai joje, kaip daugelyje kitų serveryje veikiančių programų, nenaudojami prieigos ir komandų klavišai.

Išanalizavus „Mozilla Firefox“ naršyklės lokalizuojamųjų eilučių ilgius, paaiškėjo, kad eilučių ilgis svyruoja nuo 1 ženklo iki 549 ženklų (angliškoji versija) ir nuo 1 ženklo iki 633 ženklų (sulietuvinta versija). Atmetus prieigos ir komandos klavišus, kurie yra vieno ženklo ilgio, lieka 4839 eilutės, jas vadinsime reikšminėmis eilutėmis. Reikšminių eilučių vidutinis ilgis yra 29,7 ženklai. 72 proc. reikšminių eilučių yra ne ilgesnės kaip 30 ženklų (t. y. 2–4 vidutinio ilgio žodžiai). Atsižvelgus į eilučių



35 pav. Lokalizuojamųjų tekstinių išteklių eilučių pasiskirstymas remiantis „Moodle“ pavyzdžiu

30 lentelė. Eilučių ilgių pasiskirstymas programų lokalizuojamuose ištekliuose

Eilučių ilgis, ženklais	Mozilla Firefox	Moodle	Lemill	Vidurkis
⩽ 10	33,6 %	20,8 %	23,6 %	26,0 %
⩽ 15	47,6 %	41,1 %	44,0 %	44,2 %
⩽ 20	60,0 %	37,0 %	59,7 %	52,2 %
⩽ 25	66,2 %	66,0 %	68,6 %	66,9 %
⩽ 30	72,0 %	72,0 %	74,4 %	72,8 %
⩽ 35	75,4 %	76,3 %	78,2 %	76,6 %
Vidutinis ilgis, ženklais	29,7	37,0	30,2	32,3

skaičiaus pasiskirstymą pagal ženklų skaičių gauta, kad dažniausiai pasitaiko 3–18 ženklų ilgio eilutės, t. y. nuo vieno iki trijų keturių žodžių eilutės.

Išanalizavus virtualiosios mokymosi aplinkos „Moodle“ lokalizuojamuosius išteklius, paaiškėjo, kad eilučių ilgiai svyruoja nuo 1 ženklo iki 1041 ženklų. Reikšminių eilučių vidutinis ilgis yra 37 ženklai (5–8 žodžiai). 41 proc. visų eilučių yra ne ilgesnės kaip 15 ženklų (1–3 žodžiai). 20 proc. visų eilučių yra ne ilgesnės kaip 10 ženklų (vidutiniškai 1–2 žodžiai).

Panašūs skaičiai gauti ištyrus sistemos „Lemill“ (sistema skirta mokymosi išteklių kūrimui bendradarbiaujant) lokalizuojamuosius išteklius.

Apibendrinti trijų programų lokalizuojamųjų išteklių eilučių ilgiai pateikiami 30 lentelėje.

Maždaug pusė visų lokalizuotinų eilučių yra iki 20 ženklų ilgio. Tai atitinka eilutes-frazes, kuriose yra nuo 1 iki 4 žodžių. Apie 20 proc. visų lokalizuotinų eilučių yra iki 10 ženklų ilgio – tai 1–2 žodžių eilutė (žodis arba trumpa frazė). Skaičiai gauti remiantis trijų tirtų programų pavyzdžiais, bet panašus santykis būtų gautas ištyrus ir daugiau programų. Taigi lokalizuojamuose ištekliuose dominuoja trumpos eilutės, kuriose nėra konteksto, padedančio taikliai jas išversti.

Lokalizuojamos eilutės lokalizuojamųjų išteklių failų viduje dažniausiai pateikiamos iš eilės (tiesiškai), nenurodant arba iš dalies nurodant realiai egzistuojančius sąryšius su kitomis eilutėmis ir naudotojo grafinės sąsajos elementais, kuriuose eilutės bus vaizduojamos programai veikiant (5.2 skyr.).

Dėl minėtų veiksnių programos dialogo tekstų vertimo ir adaptavimo sąnaudos yra kelis kartus didesnės, palyginus su paprasto rišlaus teksto vertimu, arba lokalizavimo kokybė nepakankama.

Visų šių nesklandumų būtų galima išvengti, jei lokalizuotojas turėtų kontekstinę informaciją. Beveik visi formatai turi priemonių komentarams rašyti. Tačiau komentavimas nėra privalomas.

„Microsoft“ ir kiti programinės įrangos gamintojai prie išteklių eilučių prirašo jų paskirtį arba vietą programos lange (pvz., mygtuko pavadinimas, meniu punktas, pranešimas apie klaidą). Tačiau nėra priemonių nurodyti ryšiams tarp eilučių, t. y. kontekstui nurodyti. Konteksto trūkumo problemas iš dalies bandoma spręsti taikant anksčiau nagrinėtą XLIFF failų formatą, bet šio formato priemonės išsamesniam kontekstui nurodyti yra ribotos. Galimas problemos sprendimas – formalizuoti kontekstinę informaciją.

5.4. Kontekstinės informacijos formalizavimas

Pateikiame vieną iš galimų kontekstinės informacijos apie išteklius įtraukimo metodų, kuriame eilučių pateikimas atspindėtų ryšius su grafinės sąsajos elementais, kuriuose jos naudojamos, o eilutės turėtų atributus, kurie suteiktų lokalizavimui naudingos informacijos, kas padėtų gerinti lokalizacijų kokybę. Metodas paremtas formaliosiomis atributinėmis gramatikomis.

Pagrindinės priežastys, dėl kurių nebuvo pasirinkta viena iš egzistuojančių grafinės sąsajos aprašo kalbų (šiuo metu daugėja XML pagrindu sukurtų daugiaplatformių sąsajos su naudotoju aprašo kalbų (Souchon, Vanderdonckt, 2003)) yra šios:

1. Grafinės sąsajos aprašo kalbos, kurios yra nepriklausomos nuo programavimo ar ženklinimo kalbų bei platformų, nėra specialiai pritaikytos internacionalizavimui ir lokalizavimui, t. y. jose nenumatyti atributai, kurie galėtų būti naudingi lokalizavimui (Souchon, Vanderdonckt, 2003).
2. Grafinės sąsajos aprašo kalbų naudojimas dar nėra pakankamai paplitęs (iš analizuotų programų tik „Mozillos“ šeimos programos naudoja savo atvirąją grafinės sąsajos aprašo kalbą XUL (Boswell, 2002).

Čia vengiama prisirišti prie konkretaus formato. Siekiama nepriklausomo formalaus išteklių struktūrizavimo ir pateikimo būdo, kuris tiktų taikyti prie įvairių šiuo metu egzistuojančių formatų. Sukurto formalaus pateikimo būdo konvertavimas į kurį nors iš formatų (pvz., XML), yra techninis konkrečios realizacijos dalykas.

5.4.1. Formaliosios ir atributinės gramatikos

Jeigu programoje būtų naudojama tam tikra grafinės naudotojo sąsajos aprašo kalba (ateityje sukurta arba viena iš esamų), tai formaliąją gramatiką kurti būtų paprasčiau, užtektų esamą formalųjį sąsajos aprašą papildyti lokalizavimui reikšmingais simboliais ir atributais, pateiktais šiame skyriuje, aprašyti semantikos taisykles.

Pradėsime nuo formaliųjų gramatikų ir atributinių gramatikų apibrėžimo ir programos išteklių dalių metainformacijos formalizavimo. Idėja gali būti išplečiama ir į kitas lokalizuojamųjų išteklių ir grafinės sąsajos sritis.

Lokalizuojamiems ištekliams ir jų metainformacijai formalizuoti galėtų tikti bekontekstės arba kontekstinės formaliosios gramatikos (31 lentelė).

31 lentelė. Gramatikos, kurios galėtų tikti lokalizuojamiems ištekliams pateikti

Gramatika	**Išvedimo taisyklių ($\alpha \to \beta$) pavidalas**	**Ribojimas**	**Atitikmuo algoritmų teorijoje**
Kontekstinė	$uAv \to uwv$	$\lvert\alpha\rvert \leqslant \lvert\beta\rvert$	Tiuringo mašina
Bekontekstė	$A \to w$	$\lvert\alpha\rvert = 1$	Dėklas

Teoriškai uždavinys būtų išsprendžiamas naudojant kontekstines gramatikas. Tačiau jų realizacija yra daug sudėtingesnė, negu bekontekstinių gramatikų. Programavimo kalbų sintaksei formaliai aprašyti praktiškai naudojamos bekontekstės gramatikos. Jos gerai ištirtos, sukurta daug jomis aprašytų programavimo kalbų.

Šiame darbe pasirinktos atributinės gramatikos, kurios yra bekontekstinių gramatikų išplėtimas, t. y. įvedamų atributų ir semantikos taisyklių dėka gaunamas tarpinis variantas tarp kontekstinių ir bekontekstinių gramatikų.

Atributinių gramatikų formalizmo pradinis tikslas buvo programavimo kalbų ir jų semantinių aspektų formalizavimas (Knuth, 1968), todėl atributinių gramatikų pagrindinė ir plačiausia tyrimų ir taikymų sritis – kompiliatorių projektavimas. Jos intensyviai nagrinėjamos ir koncepciniu, ir praktiniu požiūriais. Yra žinomi keli atributinių gramatikų poklasiai, turintys savitus realizacijos algoritmus. Taikant atributines gramatikas sukurta įvairių sistemų: metakompiliatorių, kompiliatorių, derintuvių. Kiekviena tokia sistema turi savo specifikavimo kalbą, kuri gali būti laikoma atributinės gramatikos dialektu.

Nepaisant to, kad atributinių gramatikų pradinė ir geriausiai ištirta taikymo sritis yra (tekstinės) programavimo kalbos, jų idėja gali būti kūrybiškai taikoma ir kitose srityse, kur yra svarbūs ryšiai tarp struktūrizuotos informacijos elementų (Paaki, 1995). Pastaruųjų dešimtmečių publikacijose pateikiami atributinių gramatikų

sėkmingo taikymo pavyzdžiai tokiose srityse, kaip bendroji programinės įrangos inžinerija (Shinoda, Katayama, 1988; Frost, 1992; Lewi ir kt., 1992), paskirstytasis programavimas (Kaiser, Kaplan, 1993), loginis programavimas (Deransart, Maluszynski, 1993), programų statinė analizė (Horwitz, Reps, 1992), duomenų bazės ir dokumentų užklausų formavimas (Neven, 2000), vizualusis programavimas (Crimi ir kt., 1990), natūralios kalbos naudotojo sąsajos projektavimas (Alexin ir kt., 1990), aparatinės kompiuterių įrangos projektavimas (Jones, Simon, 1986), kompiuterių komunikacijos protokolai (Van de Burgt, Tilanus, 1989), pirkėjo modelio kūrimas elektroninėse parduotuvėse (Manousopoulou, Papakonstantinou, 2000), XML semantika (Ramalho ir kt., 1998; Psaila, Crespi-Reghizzi, 1999) ir kt.

Kadangi programinei įrangai lokalizuoti skirti ištekliai gali būti laikomi struktūrizuota informacija (pvz., paėmus struktūros pagrindu ryšius tarp eilučių ir grafinės sąsajos elementų), tai dar vienas naujas atributinių gramatikų pritaikymas būtų lokalizuojamųjų išteklių struktūrizuoto semantinio pateikimo formato kūrimas.

Atributinė gramatika AG = <G, A, R> sudaroma iš trijų komponentų (Knuth, 1968):

1. Bekontekstė gramatika G.
2. Baigtinė atributų aibė A.
3. Baigtinė semantikos taisyklių aibė R.
   Čia bekontekstė gramatika G = <N, T, P, S>:
4. N – neterminalinių simbolių baigtinė aibė;
5. T – terminalinių simbolių baigtinė aibė (kalbos abėcėlė);
6. P – gramatikos taisyklių baigtinė aibė. Bendru atveju taisyklė išreiškiama pavidalu $X \to \alpha$, čia $X \in N$, $\alpha \in (N \cup T)^*$. V^* – aibė visų eilučių, sudarytų iš abėcėlės V simbolių, įskaitant ir tuščią eilutę (ε). Visų abėcėlės V eilučių, išskyrus tuščią, aibę žymėsime $V+$. T. y. išvedimo taisyklės kairėje dalyje yra vienas neterminalinis simbolis, o dešinėje dalyje – terminalinių ir neterminalinių simbolių eilutė.
7. S ($S \in N$) – pradinis neterminalinis simbolis, nuo kurio pradedamas eilučių generavimas.

Elementas, priklausantis aibei $V = N \cup T$, vadinamas gramatikos simboliu.

Su kiekvienu gramatikos simboliu $X \in V$ susiejama baigtinė atributų aibė A(X). A(X) aibė dalijama į du poaibius: paveldėti atributai I(X) ir sintezuoti atributai S(X), taip kad $I(X) \cap S(X) = Ø$. Visų gramatikos atributų aibė $A = \cup A(X)$.

Tokia yra bendra AG apibrėžtis, kuri vienoda įvairiose atributinių gramatikų taikymams skirtose publikacijose, o sintezuotų ir paveldėtų atributų aibės bei semantikos taisyklių aibės apibrėžtys įvairiuose šaltiniuose, skirtuose atributinėms gra-

matikoms ir jų taikymams nagrinėti, skiriasi. Originaliame Knutho AG apibrėžime (Knuth, 1968) laikoma, kad terminaliniai simboliai gali turėti paveldėtus atributus, tačiau negali turėti sintezuotų atributų. Ši taisyklė buvo pakeista daugumoje atributinių gramatikų realizacijų. Dažniausiai skirtingų autorių darbuose skiriasi pradinio ir terminalinių gramatikos simbolių paveldėtų ir sintezuotų atributų taisyklės.

Vėlesniuose straipsniuose dominuoja susitarimas, kad gramatikos pradinis simbolis ir terminaliniai simboliai neturi turėti paveldėtų atributų (Paaki, 1995; Reghizzi, 2009). Pagrindinė priežastis, kodėl terminaliniams simboliams priskiriami tik sintezuoti atributai, – tai modulinis transliatorių projektavimas. Norint komponuoti didesnę gramatiką iš smulkesnių dalių yra svarbu, kad terminaliniai simboliai būtų nepriklausomi nuo konteksto, tuomet leksikos analizatoriai gali būti projektuojami nepriklausomai nuo kitų komponentų.

Laikysime, kad išvedimo taisyklė $p \in P$, $p: X_0 \to X_1 \ldots X_n$ $(n \geqslant 0)$ turi atributo $X_i.a$ įeitį, jeigu $a \in A(X_i)$, $0 \leqslant i \leqslant n$.

Semantikos taisyklių baigtinė aibė yra susieta su išvedimo taisykle p. Pateikiama po vieną taisyklę kiekvienam sintezuotam atributui $X_0.a$ ir po vieną taisyklę kiekvienam paveldėtam atributui $X_i.a$, $1 \leqslant i \leqslant n$. Taigi R_p yra taisyklių rinkinys, turintis pavidalą $X_i.a = f(y_1, \ldots, y_k)$, $k \geqslant 0$, čia:

- arba $i = 0$ ir $a \in S(X_i)$, arba $1 \leqslant i \leqslant n$ ir $a \in I(X_i)$;
- kiekvienas y_j, $1 \leqslant j \leqslant k$, yra atributo atvejis išvedimo taisyklėje p;
- f yra funkcija, vadinama semantikos funkcija, atvaizduojanti $y_1, \ldots, y_k$ reikšmes į $X_i.a$ reikšmę. Taisyklėje $X_i.a = f(y_1, \ldots, y_k)$ $X_i.a$ atributo įeitis priklauso nuo kiekvieno y_j atributo įeities, $1 \leqslant j \leqslant k$. $R = \cup R_p$.

Pagal šią apibrėžtį sintezuoti atributai yra simbolių, esančių išvedimo taisyklių kairėje pusėje, išvestis, o paveldėti atributai yra simbolių, esančių išvedimo taisyklių dešinėje pusėje, išvestis. (Laikoma, kad terminalinių simbolių sintezuoti atributai apibrėžiami išorėje.)

Struktūros St, generuotos gramatikos G, *išvedimo medis* – tai medis, kuriame:

- kiekvienas mazgas yra pažymėtas simboliu $X \in V$ arba tuščia eilute ε;
- medžio šaknis žymima pradiniu simboliu S;
- jei mazgas, pažymėtas X, turi dukterinių mazgų, pažymėtų $X_1, \ldots, X_n$, tai $X \to X_1, \ldots, X_n$ yra išvedimo taisyklė iš aibės P;
- lapų simboliai, sujungti iš kairės į dešinę, formuoja struktūrą St (programavimo kalbų projektavimo atveju – programą, programinės įrangos lokalizavimo atveju – išteklių visumą, išteklyną).

St struktūros *atributinis medis* – tai jos išvedimo medis, kuriame kiekvienas mazgas n, pažymėtas simboliu X, turi jam priskirtas atributų reikšmes, atitinkančias X atributus. Tai medis, gautas atlikus atributų skaičiavimą.

Atributų skaičiavimas – tai procesas, kurio metu skaičiuojamos atributų reikšmės atributiniame medyje T, remiantis semantikos taisyklėmis R. T. y. atributo atvejo n.a, atitinkančio simbolio Y atributą a, reikšmė skaičiuojama pagal semantikos taisyklę $Y.a = f(y_1, ..., y_k) \in R_p$, čia arba (1) $a \in I(Y)$ ir p yra išvedimo taisyklė $X \rightarrow ... Y ...$, taikoma generuojant T iš n, arba (2) $a \in S(Y)$ ir p yra išvedimo taisyklė $Y \rightarrow ...$, taikoma generuojant T iš n. Semantikos taisyklė taikoma kreipiantis į funkciją f su argumentais – reikšmėmis tų atributų, kurie atitinka $y_1, ..., y_k$, ir priskiriant tos funkcijos reikšmę atributui n.a. Struktūros St prasmė yra sudaryta iš (sintezuotų) atributų, susietų su struktūros St šakniniu mazgu (toks principas taikomas klasikinėse atributinėse gramatikose).

Kadangi taikysime atributines gramatikas ne programavimo kalbos kompiliatoriui projektuoti, kur minėtas reikalavimas yra svarbus, o lokalizuojamiems ištekliams su metainformacija pateikti, imant struktūros pagrindu grafinę sąsają, tai originalaus atributinių gramatikų apibrėžimo reikalavimas, kad visa galutinė semantinė informacija renkama šakniniame medžio mazge, netenka prasmės. Dirbant su lokalizuojamaisiais ištekliais svarbus ne automatizavimas, o atributų priskyrimas ir jų pateikimas bei analizė išteklių lokalizavimo metu.

Toliau aptarsime bendruosius atributinių gramatikų lokalizuojamiems ištekliams aprašyti sudarymo principus.

5.4.2. Lokalizuojamųjų išteklių formalios gramatikos sudarymo principai

Minėjome (4.2.8 skyr.), kad lokalizavimo metu nemažai problemų kelia parametrizuotos lokalizavimui skirtos teksto eilutės ir ekrane matomos eilutės, sudarytos sujungiant (suduriant) kelias išteklių eilutes. Dėl to sudarant gramatiką lokalizuojamųjų išteklių eilutę tikslinga išskaidyti į segmentus, kuriuos skiria parametrai (jei yra). Jeigu eilutės yra suduriamos, tai gramatikos medyje jos atsiranda greta: pateikiamos kaip visą eilutę įvardijančio simbolio mazgo žemesnio lygio mazgai. Dėl to reikalingas atskiras neterminalinis simbolis grafinės sąsajos elementui nurodyti ir atskiras neterminalinis simbolis visai eilutei nusakyti.

Lokalizuojamųjų išteklių formalizavimas pradedamas nuo bekontekstės gramatikos kūrimo. Žemiau pateiksime tokios gramatikos kūrimo veiksmus.

1. Bekontekstė gramatika G = <N, T, P, S> sudaroma konkrečiai programai, atspindint jos grafinės sąsajos struktūrą ir siejant ją su lokalizuotinomis eilutėmis.
2. Gramatika pagal jos simbolių naudojimą sudaroma iš dviejų pagrindinių dalių: grafinės sąsajos struktūros ir lokalizuojamų eilučių ir jų struktūros.
3. Neterminalinių simbolių naudojimas:
   a. Grafinės sąsajos kiekvienam elementui įvardyti (pvz., mygtukui, išskleidžiamajam sąrašui, žymimajam langeliui) neterminaliniai simboliai parenkami remiantis programos grafinės naudotojo sąsajos specifika.
   b. Parenkamas neterminalinis simbolis visai eilutei iš lokalizuojamųjų išteklių įvardyti.
   c. Jei išteklių eilutėje pavartotas parametras, tai jam naudojamas neterminalinis simbolis, o išvedimo taisyklėje, kurios kairėje yra parametro neterminalinis simbolis, dešinėje pusėje yra vienas ar keli neterminaliniai simboliai, žymintys eilutes, skirtas įrašyti vietoje parametro, arba ε, jeigu parametro reikšmė skaičiuojama (gaunama) dinamiškai programos vykdymo metu. Lokalizuojamųjų išteklių eilutėje parametras paprastai žymimas taip, kaip priimta naudojamame lokalizuojamųjų išteklių atskyrimo metode arba programavimo kalboje, kuria parašyta programinė įranga, pavyzdžiui, %S, #1, #3, @vardas. Sudarant gramatiką, šis vardas pakeičiamas neterminaliniu simboliu, reiškiančiu parametrą.
   d. Jei meniu ar kitame grafinės sąsajos elemente (valdiklyje) rodomas tekstas, suduriamas iš kelių eilučių, tai gramatikos išvedimo medyje jos atsiduria greta. Tam įvedami atskiri neterminaliniai simboliai grafinės sąsajos elementui ir visai eilutei.
4. Terminaliniai simboliai – tai lokalizuojamos eilutės ar eilučių dalys (čia eilutės dalis vadinsime segmentais). Jei eilutėje nėra parametrų, tai visa eilutė atitinka terminalinį simbolį. Jei eilutėje yra parametrų, tai eilutė skaidoma į segmentus, kuriuos skiria parametrai.

Sukūrus programos lokalizuojamiems tekstiniams ištekliams aprašyti skirtą bekontekstę gramatiką, tolesniais žingsniais ji yra išplečiama iki atributinės gramatikos:

5. Gramatikos simboliams priskiriami atributai, skirti lokalizavimo požiūriu svarbiai semantinei informacijai nusakyti. Visą eilutę atitinkančiam neterminaliniam simboliui priskiriamas atributas, apibūdinantis visą eilutę, t. y. eilutės aprašas. Atributų parinkimas remiasi lokalizavimo klaidų analize ir kalbos ypatumais.
6. Apibrėžiamos taisyklės kiekvieno gramatikos simbolio atributų reikšmėms skaičiuoti.

Pastebėsime, kad grafinė sąsaja, kuria remiantis sudaroma atributinė gramatika, čia supaprastinama iki vaizdo, kuris veikiant programai bus rodomas kompiuterio ekrane. Į aprašą įtraukiami visi elementai, kurie nėra neutralūs lokalizavimo požiūriu. Nepateikiame grafinių elementų tarpusavio sąveikos aprašų ir sąveikos realizacijos priemonių, nes tai nėra reikšminga lokalizavimo požiūriu ir tik darytų formalųjį aprašą sudėtingesnį. Tie sąveikos elementai, kurie teikia svarbios informacijos lokalizuotojui, įtraukiami per atributus, pavyzdžiui, išskleidžiamojo sąrašo elementų eilė, ryšys su komanda, realizuojančia vidinę funkciją. Į aprašą neįtraukiami tie elementai, kuriuose nėra vaizduojama lokalizuotina eilutė ar lokalizavimo požiūriu nesvarbi informacija, pvz., pelės grafinis žymeklis.

Kituose skyreliuose išsamiau aptarsime gramatikos simbolių parinkimą, gramatikos išvedimo taisyklių sudarymą ir išplėtimą iki atributinių gramatikų.

5.4.3. Grafinės sąsajos elementų įtraukimas į lokalizuojamųjų išteklių gramatiką

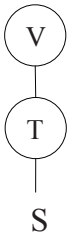
Lokalizuojamųjų išteklių ir jų metainformacijos formalus aprašas pradedamas kurti nuo bekontekstės gramatikos, kuri sudaroma remiantis grafinės sąsajos elementų struktūra, pagal kurią galima atsekti, kur lokalizuojamoji eilutė bus rodoma grafinėje sąsajoje. Išskirsime pagrindinius atvejus, kaip lokalizuojamos eilutės gali būti rodomos grafinės sąsajos elementuose (valdikliuose) – nuo to priklausys gramatikos simbolių parinkimas ir išvedimo taisyklių sudarymas. Vaizdumo dėlei naudosime grafinį gramatikos simbolių parinkimo ir struktūros vaizdavimą (gramatikos išvedimo medžio atvejų fragmentus), tolesniuose skyreliuose gramatikos simbolių ryšius aprašysime išvedimo taisyklėmis.

Ištisa eilutė paprastame valdiklyje. Eilutė be parametro (vienas segmentas S, mazgas V atitinka valdiklio neterminalinį simbolį, mazgas T – visos teksto eilutės neterminalinį simbolį) (36 pav.).

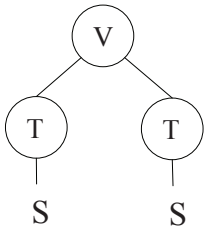
Viename paprastajame valdiklyje rodomos sudurtos eilutės. Kiekviena tokia eilutė turi savo vardą lokalizuojamuose ištekliuose (t. y. eilutės pateikiamos atskirai). Toks būdas lokalizavimo praktikoje pasitaiko gana dažnai, bet yra nepageidautinas, nes gali sukelti klaidų lokalizuojant programas tam tikroms kalboms, ypač, kai reikia derinti suduriamų eilučių gramatines formas (4.2.8 skyr.). Teoriškai galimas atvejis, kad suduriamos eilutės turės parametrų. Praktiškai toks atvejis pasitaiko retai, nes paprastai eilučių sudūrimas pakeičia parametro funkciją. Toliau pateiktas dviejų suduriamų eilučių be parametrų variantas (alternatyva) (37 pav.).

Lokalizavimui skirtoje eilutėje pavartoti statiniai parametrai. Galimi atvejai:

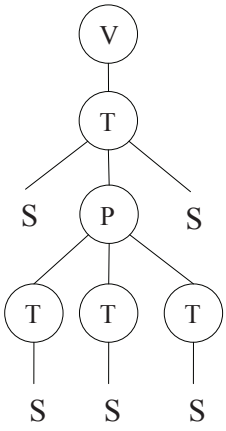
1. Vietoj parametro įrašoma kita, viena iš kelių galimų eilučių iš lokalizuojamųjų išteklių. Laikysime, kad parametrą pakeičia eilutė be parametro. Tuomet eilučių grupę, iš kurios programos vykdymo metu pasirenkama eilutė įrašyti vietoje parametro, žymėsime neterminaliniu simboliu P (38 pav.).
2. Parametro vietoje įrašoma reikšmė, kurios nėra lokalizuojamuose ištekliuose, ji sužinoma programos vykdymo metu. Pavyzdžiui, kokių nors objektų skaičius, naudotojo vardas, pavardė ir pan. Tuomet reikalingas parametro atributas, suteikiantis informacijos lokalizuotojui, kokio tipo duomenys bus įrašomi vietoje parametro programos vykdymo metu. 39 paveiksle pavaizduotas variantas, kai eilutės viduje yra vienas parametras su dinamiškai parenkama reikšme.



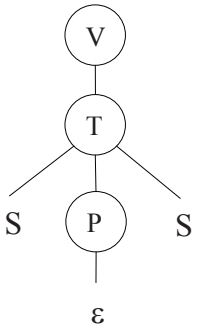
36 pav. Ištisa eilutė paprastame valdiklyje



37 pav. Sudurtos eilutės paprastame valdiklyje



38 pav. Eilutė su parametru, vietoj kurio įrašoma viena iš eilučių iš lokalizuojamųjų išteklių



39 pav. Eilutė su parametru, vietoj kurio įrašoma reikšmė programos vykdymo metu

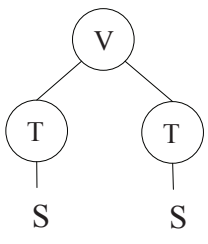
Valdiklyje rodoma viena iš kelių eilučių, priklausomai nuo konteksto, kuris paaiškėja vykdant programą (40 pav.).

Toks būdas dažnai taikomas, kai programos kūrėjai bando išvengti parametrų naudojimo lokalizuotinose eilutėse: kiekvienam konteksto atvejui sukuriama atskira eilutė. Šis variantas įtraukiant į gramatiką turi tokią pačią struktūrą, kaip ir 2-as variantas (viename paprastajame valdiklyje rodomos sudurtos eilutės), todėl papildoma informacija, ar eilutės yra suduriamos, ar parenkamos dinamiškai rodant vieną iš jų, pateikiama naudojantis atributais.

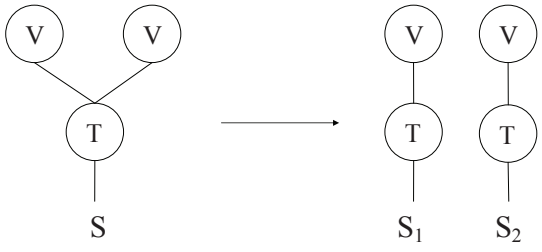
Išteklių eilutė naudojama keliuose naudotojo grafinės sąsajos elementuose (pvz., mygtuko tekstas ir kontekstinio meniu elementas). Tokia eilutė būtų dubliuojama, lyginami atributai. Jei dubliuotų eilučių atitinkami atributai dera tarpusavyje, tai ištekliuose gali likti viena eilutė. Jei atributai nedera, tai reiškia, kad viename valdiklyje reikia vienokio vertimo, kitame – kitokio. Šitaip aptinkamos internacionalizavimo klaidos. Jas galima ištaisyti į lokalizuojamus išteklius įtraukiant atskiras eilutes (41 pav.).

Šiuo atveju gramatika taip pat padėtų minimizuoti eilučių skaičių. Kartais programinės įrangos kūrėjai, siekdami išvengti aukščiau minėtos internacionalizavimo klaidos, įtraukia į išteklius per daug tos pačios eilutės egzempliorių. Sudarius atributinę gramatiką ir nagrinėjant atributus būtų galima nustatyti, ar reikia atskiro eilutės egzemplioriaus, ar ne.

Sudėtinis valdiklis (t. y. valdiklis, kuriame rodomos kelios eilutės), pavyzdžiui, išskleidžiamasis ar kitoks sąrašas. Programos meniu taip pat yra sudėtinis valdiklis, tačiau apie jį kalbėsime atskirai (jis yra sudėtingesnis, turi eilučių hierarchiją, savo lokalizavimo ypatumų). Nagrinėjant sudėtinio valdiklio kiekvieną paprastą elementą E (42 pav.), jam gali būti taikomas vienas iš anksčiau aprašytų atvejų. Paveiksle pavaizduotas variantas, kai sudėtinis valdiklis turi tris paprastus elementus (E), viename iš jų pavartota eilutė su parametru.

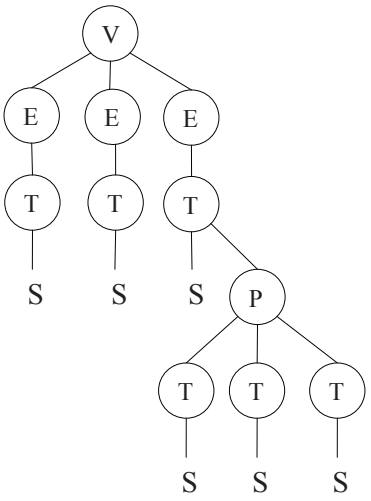


40 pav. Eilučių alternatyvos paprastame valdiklyje



41 pav. Ta pati eilutė keliuose grafinės sąsajos elementuose

Neterminalinis simbolis T, reiškiantis visą eilutę iš išteklių sistemos, įvestas laikantis vienodo bendro gramatikos sudarymo principo, kadangi teoriškai viename paprastajame elemente (E) gali būti vaizduojamos kelios eilutės iš lokalizuojamųjų išteklių aibės (viena greta kitos arba kaip alternatyvos).

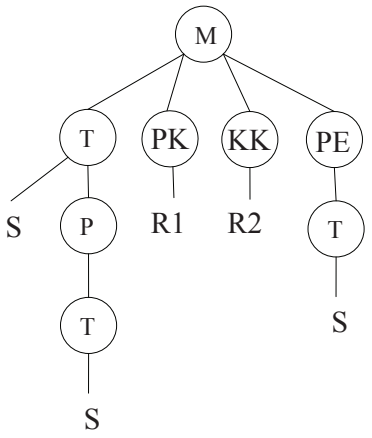


42 pav. Sudėtinis valdiklis

Programų, veikiančių klientų kompiuteriuose, grafine sąsaja pateikiamos komandos paprastai turi prieigos klavišus (pvz., skirtą iškviesti komandą paspaudus tokį klavišą kartu su alternatyvos klavišu, veikia tik tam tikrame kontekste, pvz., tuo metu atvertame lange) bei komandos klavišus (kurį paspaudus su vienu ar daugiau kitų klavišų (pvz., valdymo) vykdoma tam tikra komanda, nesvarbu koks kontekstas, t. y. kuris langas tuo metu atvertas arba kur yra žymeklis), taip pat komandą paaiškinančią etiketę. Pavyzdžiui, naršyklės „Mozilla Firefox“ prieigos ir komandų klavišai sudaro 17 proc. visų lokalizuojamųjų išteklių. Prieigos ir komandų klavišams stengiamasi parinkti labiausiai įsimenamą raidę (Dagienė ir kt., 2008), pavyzdžiui, prieigos klavišui geriausiai tinka pirmoji komandos ar meniu punkto pavadinimo raidė. Jeigu toje pačioje pakopoje yra keli meniu punktai, prasidedantys ta pačia raide, tai tada tenka pasirinkti kitą raidę: kito žodžio pirmąją (jeigu pavadinimas iš kelių žodžių), antrojo skiemens pirmąją raidę, kirčiuotą raidę ir pan. Vengiama raidžių, turinčių žemiau bazinės linijos išsikišusių dalių (ą, ę, g, į, j, ų), kad su jomis nesusilietų pabraukimo brūkšnys. Komandos klavišui taip pat geriausiai tinka arba pirmoji komandos raidė, arba originali raidė (jei toks klavišas jau yra prigijęs, pavyzdžiui, daugumoje tekstų rengyklių *Vald* + *C* reiškia kopijavimą).

Kadangi lokalizavimo metu komandų pavadinimai (eilutės) keičiami, tai turi būti pakeisti ir prieigos bei komandos klavišai. Todėl jie iškeliami į lokalizuojamuosius išteklius. Išanalizavus daugelio programų lokalizuojamuosius išteklius galima padaryti išvadą, kad šiuo metu naudojami du pagrindiniai tokių klavišų pateikimo ištekliuose būdai:

1. Pateikiami kaip atskiros įvardytos eilutės (pvz., „Mozillos“ šeimos programose: *button.accesskey “E”*, *button.commandkey “E”*);



43 pav. Prieigos komandos klavišai ir paaiškinančios etiketės

2. Prieigos klavišai kaip nors pažymimi eilutės viduje, dažniausiai ženklu & (pvz., *&Edit image*), o komandos klavišai rašomi po komandos pavadinimo (pvz., *&Edit image <F4>*).

Valdikliai gali turėti paaiškinančiųjų etikečių (valdiklį paaiškinantis užrašas, parodomas atvedus ant valdiklio žymeklį). Lokalizuojamuose ištekliuose toks paaiškinimas pateikiamas kaip atskira eilutė.

Sudarant gramatiką siekiama, kad visos lokalizuojamųjų išteklių eilutės, susijusios su tam tikru valdikliu, taisyklėje būtų pateikiamos taip, kad valdiklį žymintis neterminalinis simbolis būtų išvedimo taisyklės kairėje, o visą valdiklio pagrindinę eilutę, prieigos klavišą, komandos klavišą, paaiškinančiąją etiketę žymintys neterminaliniai simboliai būtų išvedimo taisyklės dešinėje. Tipinio mygtuko su lokalizuotinomis eilutėmis gramatikos išvedimo medžio fragmentas galėtų atrodyti taip (43 pav.):

Čia M – mygtukas, T – visa teksto eilutė (mygtuko atliekamos komandos pavadinimas), PK – prieigos klavišas (R1 – konkreti raidė, žyminti šį klavišą), KK – komandos klavišas (R2 – konkretus simbolis, žymintis tą klavišą), PE – paaiškinančioji etiketė, P – parametras, S – terminalinis simbolis (eilutės segmentas arba visa eilutė).

Prieigos klavišus, komandų klavišus ir paaiškinančias etiketes išskyrėme tam atvejui, kai programose jos numatytos ir naudojamos. Dalis programų tokių naudotojo sąsajos elementų neturi, tuomet struktūra būtų paprastesnė (be PK, KK, PE ir jų išvestinių mazgų).

5.4.4. Gramatikos simbolių ir lokalizuojamųjų išteklių elementų santykis

Tekstinių lokalizuojamųjų išteklių eilučių aibę E galima suskirstyti į dvi dalis:

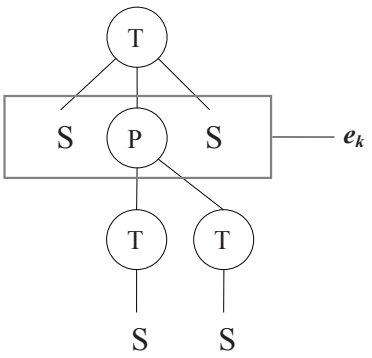
1. Tikrosios teksto eilutės, kurios skirtos rodyti ekrane (įskaitant eilutes su parametrais), tokių eilučių aibę žymėsime E_T.

2. Su lokale susijusias programos nuostatas apibrėžiančios eilutės (pvz., pirmoji savaitės diena, kurios reikšmė gali būti 1 (pirmadienis) arba 0 (sekmadienis); lango plotis ir aukštis; laiko formatas – 24 ar 12 valandų), tokių eilučių aibę žymėsime E_P. Tam tikrų programų atvejais $E_P = Ø$. Šios aibės elementų skaičius priklauso nuo programos projektuotojų pasirinkto sprendimo. Kai kurie autoriai laiko, kad iškelti programos nuostatas apibrėžiančias eilutes į lokalizuojamuosius išteklius nėra saugu: lokalizavimo metu dėl nepakankamos informacijos galima nurodyti netinkamą nuostatos reikšmę ir programa bus pažeista. Pavyzdžiui, mokymosi išteklių kūrimo bendradarbiavimo sistemos „Lemill“ aibė $E_P = Ø$.

Atskiro neterminalinio simbolio visai lokalizuojamųjų išteklių tekstinei eilutei pažymėti įvedimas (žymėsime juos T_i) formuoja šių simbolių aibę, kuri atitinka lokalizuojamųjų išteklių eilučių aibės E poaibį E_T. Yra tik ta išimtis, kad dalis T elementų atitiks padubliuotus aibės E_T elementus (5.4.3 skyr. 5 atvejis), o taip pat, jei eilutėje iš E_T aibės yra pavartotas parametras (kintamasis), tai vietoje jo faktinio vardo eilutėje (%S, #1, #3, @vardas ir pan.) naudojamas neterminalinis simbolis, reiškiantis parametrą (pvz., P). 44 paveiksle pavaizduotas gramatikos išvedimo medžio fragmentas, kuriame viršutinis T mazgas žymi visą eilutę, o stačiakampiu apvesti mazgai žymi atitinkamos eilutės $e_k \in E_T$ tekstą.

Eilučių, priklausančių aibei E_P, negalima tiesiogiai pamatyti kompiuterio ekrane (išimtis, kai su lokale susijusios programos nuostatos parenkamos naudojantis tam skirtais programos grafinės sąsajos elementais). Todėl natūralu tokias eilutes atvaizduoti į gramatikos simbolių atributus. Pavyzdžiui, dialogo lango plotį ir aukštį – kaip to lango neterminalinio simbolio atributus, pirmosios savaitės dienos atributą – kaip visos programos pagrindinio lango atributą.

Kiekvieno simbolio T atributų, reiškiančių eilutės identifikatorių (T.id), aibė atitinka lokalizuojamųjų išteklių aibės eilučių vardų poaibį V. Bendru atveju {T.id} ir V aibės elementai gali nesutapti, nes sudarant gramatiką gali būti vartojamas ir išplėstas eilutės identifikatorius (pvz., kelias iki



44 pav. Lokalizuojamos eilutės su parametru formalizavimas

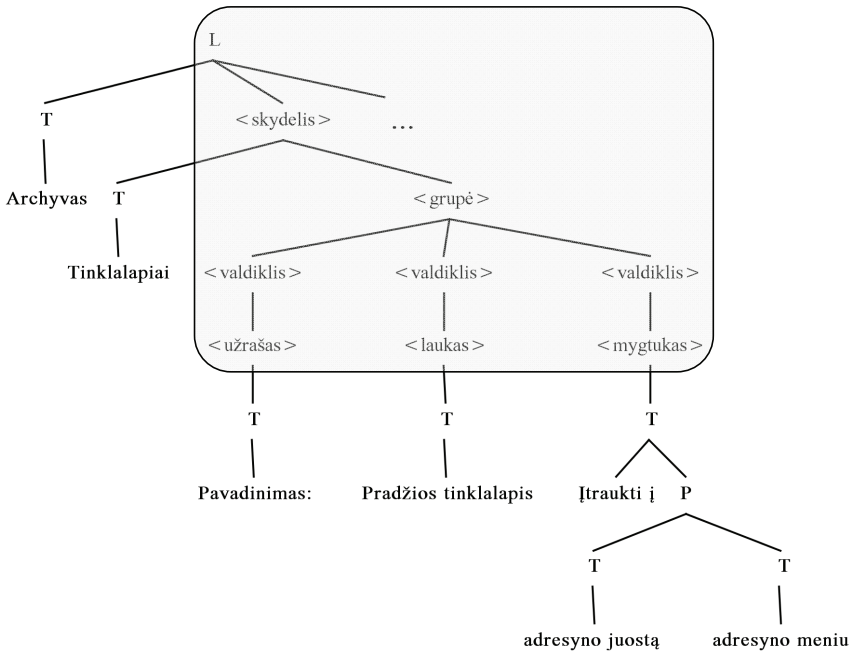
lokalizuojamųjų išteklių failo ir eilutės vardas), ne tik vardas, nurodytas lokalizuojamuose ištekliuose.

Taigi matome, kad tie duomenys, kurie pateikiami programinės įrangos tekstiniuose lokalizuojamuose ištekliuose atitinka tik vieno iš sudaromos atributinės gramatikos elementų aibę, to elemento vieno iš atributų aibę ir dalį kitų gramatikos neterminalinių simbolių atributų (jei yra).

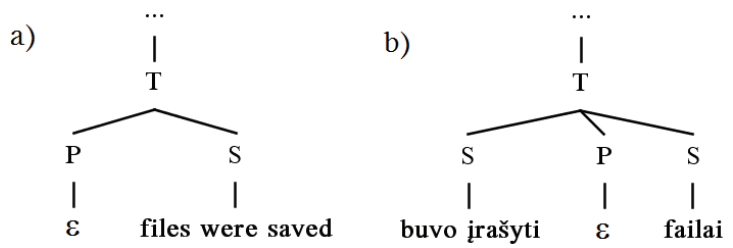
Visi kiti gramatikos simboliai ir atributai išplečia lokalizuojamuosius išteklius (laikant, kad jie pateikiami šiuo metu daugumos programinės įrangos gamintojų priimtu būdu), įtraukia struktūrinę informaciją apie grafinę sąsają, elementų semantiką ir eilutės dalis – segmentus bei parametrus.

Laikantis minėto neterminalinių simbolių parinkimo susitarimo, išteklių gramatikos išvedimo medžio šakos kiekvienas n – 1 mazgas (laikant, kad visa medžio šaka turi n mazgų) – tai T neterminalinis simbolis, atitinkantis visą teksto eilutę iš lokalizuojamųjų išteklių.

Jei gramatikos išvedimo medžio šakos n – 2 mazgas atitinka eilutės parametrą, tai lokalizuojant programą reikia nagrinėti n – 3 mazgą kartu su visais jo žemesnių



45 pav. Lokalizuojamųjų išteklių gramatikos struktūra: grafinės sąsajos ir eilučių dalys



46 pav. Simbolių skaičiaus ir eilės tvarkos pasikeitimas lietuviškoje lokalizacijoje

lygių mazgais. Šis mazgas atitinka teksto eilutę, skirtą rodyti tam tikrame valdiklyje su parametru bei vietoje jo skirtomis įrašyti eilučių alternatyvomis.

Minėtu būdu sudarytos lokalizuojamųjų išteklių gramatikos išvedimo medį galima sąlyginai suskirstyti į 2 dalis:

1. Grafinės sąsajos elementai.
2. Lokalizuojamos eilutės $\in E_T$ ir jų dalys.

45 paveiksle pavaizduotas išvedimo medžio fragmentas gramatikos, atitinkančios hipotetinės programos dialogo langą dalį ir jame skirtas rodyti 7 eilutes iš lokalizuojamųjų išteklių. Viena iš eilučių – su parametru. Parametro vietoje įrašoma viena iš galimų dviejų eilučių, esančių lokalizuojamuose ištekliuose. Apvesta ir pilkai nuspalvinta išvedimo medžio fragmento dalis atitinka gramatikos grafinės sąsajos elementų dalį, o pusjuodžiu šriftu pateikti gramatikos simboliai, atitinkantys lokalizuojamųjų eilučių dalį. Anksčiau naudotus simbolius eilutės segmentui S įvardyti čia pakeitėme konkrečiais lokalizuojamųjų eilučių tekstais.

Gramatikos dalis, atitinkanti grafinės sąsajos aprašą (45 pav. pilkas apvestas fragmentas), paprastai išlieka ta pati programinės įrangos originalui ir jos lokalizacijoms. Išimtis – kai lokalizuojant programinę įrangą modifikuojama ir jos grafinė sąsaja.

Tolesnė gramatikos dalis (mazgai T ir jų visi žemesnio lygio mazgai) yra nevienoda programinės įrangos originalui ir lokalizacijoms. Nuo simbolio T pradedamas kalbos keitimas. Lokalizuojant eilutę su parametru gali būti keičiama parametrų bei segmentų eilės tvarka, pavyzdžiui, eilutę „%S files were saved“ (laikykime, kad parametro reikšmė yra sveikasis skaičius, kuris priklauso intervalui [2; 5]) atitiktų T mazgo žemesnio lygio mazgų eilė P S (46 pav., a), o išverstą į lietuvių kalbą eilutę „buvo įrašyti %S failai“ atitiktų mazgo T žemesnio lygio mazgų eilė S P S (46 pav., b).

Todėl lokalizacijos gramatikoje gali būti skirtumų, lyginant su originalia, pradedant T simboliu link išvedimo medžio lapų.

5.4.5. Gramatikos simbolių atributai ir semantikos taisyklės

Šiuo metu egzistuojančiuose tekstinių lokalizuojamųjų išteklių pateikimo formatuose, kaip jau minėjome 5.2 skyriuje, trūksta kontekstinės informacijos apie eilutes, kuri padėtų lokalizuotojui numatyti lokalizuotinos eilutės vietą grafinėje sąsajoje, taikytiną gramatinę formą ir kitas eilutės savybes. Pagrindinis metainformacijos šaltinis šiuo metu egzistuojančiuose formatuose – tai lokalizavimo komentarai. Tačiau lokalizuotinų eilučių komentavimo praktika yra labiau panaši į išimtį negu taisyklę. Pavyzdžiui, remiantis naršyklės „Mozilla Firefox“ lokalizuojamųjų išteklių analize, tik 4 proc. eilučių turi lokalizavimo komentarus.

Kai kuriose programose dalį atributų galima išskirti iš eilučių vardų aibės elementų – vardo struktūros, pvz., *PageInfoTitle*, dalį – iš failų ir aplankų vardų (jie paprastai atspindi programos komponento pavadinimą, kuriame naudojama eilučių grupė), iš lokalizavimo komentarų. Tačiau tokias vardų schemas pasirenka ne visi programinės įrangos kūrėjai, be to, net toje pačioje programoje vardų schema nėra vieninga. Kai kurių programų eilučių vardai yra eilučių eilės numeriai, lokalizuotojui nesuteikiantys jokios informacijos apie kontekstą. Kiti formatai (pvz., „Gettext“) lokalizuotinas eilutes identifikuoja ne atskiru vardu, o pačios eilutės tekstu.

Dalį konteksto teikia eilučių suskirstymas į lokalizavimui skirtus failus (failas ar jų grupė atitinka tam tikros programų teminės dalies eilučių rinkinį). Tuomet failų vardai gali teikti tam tikros informacijos apie kontekstą, paprastai apie programos komponentą, kuriame naudojamos eilutės. Taip yra „Mozillos“ šeimos programose, virtualiojoje mokymosi aplinkoje „Moodle“, tačiau kai kurių programų autoriai sudeda visas lokalizavimui skirtas eilutes į vieną failą (pvz., „LeMill“ aplinka). Bendros taisyklės nėra.

Atlikus programinės įrangos pagrindinių kultūrinių elementų analizę ir pasirinkus tuos elementus, kurie vienaip ar kitaip atspindimi tekstiniuose lokalizuojamuose ištekliuose, galima išskirti bazinį sąrašą atributų, kuris yra svarbus sudarant programos lokalizuojamųjų išteklių atributinę gramatiką.

Dauguma pasaulyje paplitusios programinės įrangos sąsajos tekstų iš pradžių rašomi anglų kalba, o tik vėliau lokalizuojami adaptuojant kitoms rinkoms. Todėl reikia numatyti atributus, kurie padėtų atspindėti kalbos, į kurią lokalizuojama, savybes, išreikštas taip, kad originalios programinės įrangos projektuotojas, neturėdamas kitų kalbų, kurioms ruošiamasi lokalizuoti jo projektuojamą programinę įrangą, visapusiškų žinių, galėtų tokius atributus priskirti. Taigi atributus reikėtų reikšti neprisirišant prie kalbos, kuriai numatoma lokalizuoti, specifikos.

Rengiant atributų sąrašą vadovautasi programinės įrangos lokalizavimo lietuvių kalbai patirtimi (Dagienė ir kt., 2004; Dagienė, Jevsikova, 2009; Jevsikova, 2006) bei aptiktomis problemomis (4.2 skyr.). Buvo stengtasi parinkti pakankamai bendrus atributus, tinkančius daugumai abėcėlinių fleksinių kalbų. O lietuvių kalba yra viena seniausių ir palyginti daug subtilumų turinti kalba Europoje. Todėl tie patys atributai tiks ir kitoms fleksinėms abėcėlinėms Europos kalboms, nes dauguma jų turi ne daugiau fleksijų negu lietuvių kalba.

Kontekstas, kurį vaizduoja gramatikos medis ir perduoda atributai, padeda tiksliau išversti ir adaptuoti programos originalo teksto eilutes. Pavyzdžiui, angliška eilutė *File* galėtų būti verčiama vienaip, kai ji yra pagrindiniame meniu (pagal susitarimą dažniau vartojamas variantas *Failas*, tačiau kai kuriuose specializuotose programose galimi ir kiti variantai, pavyzdžiui, *Dokumentas* (jei tai dokumentų rengyklė), *Tinklalapis* (jei tai naršyklė ar tinklalapių rengyklė) ir pan.), kitaip ši eilutė galėtų būti verčiama kitose programos vietose (pvz., tai gali būti veiksmažodis *file –> įtraukti į [aplanką]*, arba *failas*, jeigu pagrindiniame meniu buvo pavartotas kitas šio meniu punkto vertimas).

Konteksto žinojimas taip pat leidžia išvengti nereikalingo žodžių kartojimo, pavyzdžiui, meniu *Failas –> Įrašyti failą*, *Failas –> Atverti failą* pakaktų pateikti taip: *Failas –> Įrašyti*, *Failas –> Atverti*.

Išskirkime universalius atributus, kurie gali būti taikomi įvairioms kalboms ir įvairioms programinės įrangos išteklių dalims lokalizuoti. Tolesniuose skyreliuose šie atributai bus papildyti specifiniais atributais, atspindinčiais konkretaus programos komponento grafinės sąsajos ir lokalizavimo savybes.

Daiktavardinė ar veiksmažodinė frazė. Šiuo atributu siekiama išvengti dažnos klaidos dėl daiktavardžių ir veiksmažodžių vienodos rašybos anglų kalboje. Ypač naudingas trumpoms eilutėms (frazėms ar net vieno žodžio eilutėms). Pvz., *Open file* arba tiesiog *Open* meniu kontekste būtų verčiamas *Atverti failą* arba *Atverti* (tai komandos pavadinimas, kuriam labiau priimtina veiksmažodinė forma), o dialogo lango pavadinime ta pati eilutė turėtų jau daiktavardinį atitikmenį *Failo atvėrimas* arba *Atvėrimas*. Galima suformuluoti reikalavimus, kada vartojama daiktavardinė frazė (pvz., programos pagrindinio meniu juostoje – pirmojo lygio komandos, langų ir kitų objektų pavadinimai), o kada veiksmažodinė (pvz., mygtuko pavadinimas, meniu komandos, esančios gilesniame negu pirmojo lygio meniu). Šis atributas yra rekomendacinio pobūdžio, kai priskiriamas automatiškai.

Valdantysis eilutės ar segmento žodis. Tai žodis, nuo kurio priklauso kitų frazės žodžių formos. Atributas ypač naudingas trumpoms eilutėms (esant ilgoms – vieno ar

kelių sakinių eilutėms šis atributas gali būti nevartojamas, kadangi kontekstą teikia pati eilutė). Keletas pavyzdžių:

Save file – Įrašyti failą (valdantysis žodis yra *save – įrašyti*),
New page – Naujas tinklalapis (valdantysis žodis yra *page – tinklalapis*),
Properties set – Nuostatų rinkinys (valdantysis žodis yra *set – rinkinys*),
Set properties – Parinkti nuostatas (valdantysis žodis yra *set – parinkti*).

Pastarųjų dviejų pavyzdžių frazėse tiksliau parinkti vertimą padėtų veiksmažodinės ar daiktavardinės frazės atributas.

Eilutės aprašas. Tai visos teksto eilutės iš lokalizuojamųjų išteklių aprašas. Šis aprašas padeda geriau suprasti eilutės paskirtį programinėje įrangoje. Ypač naudingas tada, kai dėl grafinės sąsajos elemento tekstui rodyti skiriamos vietos trūkumo eilutė sutrumpinama arba pakeičiama kita, trumpesne, bet mažiau aiškia, ir kontekstas prarandamas. Projektuotojas žino, ką turi reikšti ta eilutė, tačiau lokalizuotojai tokios informacijos neturi. Atributas padėtų užfiksuoti eilutę paaiškinančią informaciją, kuri yra svarbi parenkant taiklų vertimą, o paprastai programos eilučių vertimas būna ne pažodinis. Šis atributas taip pat labiau naudingas trumpoms eilutėms, tačiau reikalingas ir ilgesnėms eilutėms, nes gali padėti suprasti situaciją, kada ta eilutė yra rodoma (pvz., kuris nors retai pasitaikančios klaidos pranešimas).

Pirmoji eilutės raidė (didžioji ar mažoji). Programos meniu tos pačios šakos gilyn einančios komandos dažnai yra susijusios tarpusavyje, gali formuoti vieną sakinį ar frazę. Tuomet pirmos komandos eilutę vertėtų pradėti didžiąja raide, o susijusių komandų – mažosiomis, pavyzdžiui, meniu *File –> Save As –> Draft* (čia rodyklės žymi perėjimą į žemesnio lygio meniu) lietuvių kalboje turėtų atitikmenį *Failas –> Įrašyti kaip –> juodraštį*. Atributo reikšmė gramatikoje gali būti apskaičiuojama automatiškai, pvz., pagal meniu lygį. Tuomet atributo reikšmė būtų rekomendacinio pobūdžio: jei meniu lygis yra lygus 3 ar gilesnis, tai komanda gali prasidėti iš mažosios raidės, o galutinį sprendimą priima žmogus.

Anglų kalboje daugeliu atveju yra priimta sakinio viduje naudoti didžiąsias raides, pavyzdžiui, mėnesių ir dienų pavadinimus, antraštines frazes. Lietuvių kalboje, jei tokie ar panašūs žodžiai yra sakinio viduryje, jie pradedami mažąja raide. Didžioji raidė paprastai vartojama tik sakinio pradžioje, esant santrumpoms, tikriniam žodžiui arba simboliniam vardui, kuris paprastai papildomai rašomas kabutėse.

Gramatinė forma. Kadangi negalime tiesiogiai nurodyti linksnio atributo (dėl to, kad tokio linksnio atitikmens nėra anglų kalboje), kalbos dalies ir kitų specifinių atributų, tai reikėtų kitaip išreikšto atributo, kuris galėtų padėti nusakyti arba iš dalies

nusakyti žodžių gramatinę formą. Taigi žodžio ar frazės arba eilutės, skirtos įrašyti parametro vietoje, forma gali būti apibūdinama klausimu, į kurį atsako: *who*, *what*, *whom*, *where*, *when* arba prielinksniu: *to*, *from*, *in*. *Who* ir *what* klausimus galima išskirti į dvi grupes: veikiantysis objektas ir veikiamasis objektas (atitiktų lietuvių kalbos galininko linksnį). Pavyzdžiui, sakinyje *the browser cannot display this page* „the browser“ yra veikiantysis objektas, o „this page“ – veikiamasis objektas. Atributo reikšmė ir būtų šis trumpas klausimas arba prielinksnis, kuris gali iš dalies nusakyti eilutės ar jos dalies linksnį ar formą. Atributas ypač naudingas parametro simboliui, kai vietoje parametro jo reikšmė įrašoma programos vykdymo metu. Šiuo atributu nesiekiama tiksliai nusakyti bet kurį kalbos linksnį, bet pateikti informaciją, kuri padėtų lokalizuotojui pasirinkti tinkamą formą, nesupainioti veikiančiojo objekto su veikiamuoju.

Terminas. Programose pasitaiko kompiuterijos terminų, kurie anglų kalba rašomi vienodai, bet turi skirtingas reikšmes. Tos reikšmės turi skirtingus atitikmenis kitose kalbose. Keletas pavyzdžių:

- *call* – (1) skambutis, (2) kreipinys;
- *key* – (1) klavišas, (2) raktas, (3) kodas, (4) šifras;
- *tab* – (1) ąselė, (2) kortelė, (3) tabuliavimo klavišas, (4) tabuliavimo žymė;
- *release* – (1) laida, (2) atleisti, (3) išleisti;
- *manager* – (1) administratorius, (2) tvarkytuvė, (3) tvarkytojas it t. t.

Termino reikšmės numeriui nurodyti gali būti naudojamas programoje vartojamų terminų žodynas arba bendras kompiuterijos žodynas, kuriuo sutarta vadovautis lokalizuojant programą. Tam, kad programos autorius galėtų sužymėti terminų reikšmes, žodynas turi būti verčiamasis-aiškinamasis. Žodyno sudarymas gali būti vienas iš lokalizavimo darbų etapų bendradarbiaujant programos autoriams su lokalizuotojais: prieš pradedant lokalizuoti suderinama terminija, verčiamas autoriaus parengtas pagrindinių terminų žodynas.

Atsižvelgiant į tai, kokiu žodynu sutarta naudotis nurodant termino reikšmės atributą, tuo pačiu atributu gali būti nusakyta ir kalbos dalis. Pavyzdžiui, „Enciklopediniame kompiuterijos žodyne“ (Dagienė ir kt., 2008) pateikiami ir kompiuterinių programų komandų pavadinimai, kurie turi veiksmažodinę formą, taip pat būdvardžiai, nusakantys tam tikras objektų savybes. Dėl to šis žodynas nėra vadinamas terminų žodynu, jame aprašoma kompiuterijos leksika. Jei priskiriant atributų reikšmes būtų naudojamasi terminų žodynu, tai jame būtų mažiau atskirų terminų reikšmių, atitinkančių skirtingas kalbos dalis. Todėl būtų reikalingas ir anksčiau aptartas atributas, skirtas nusakyti, kokia yra frazė: daiktavardinė ar veiksmažodinė. Be to, kalbos dalį nusakantis

atributas būtų reikalingas toms frazėms, kuriose nėra nevienareikšmių terminų, t. y. termino atributui nebus priskirta reikšmės iš sutartinio žodyno.

Vidinė funkcija. Dar daugiau kontekstinės informacijos suteiktų informacija apie vidinę funkciją, kuri realizuoja lokalizuojamųjų išteklių eilute pavadintą komandą. Vienas iš pavyzdžių galėtų būti toks. Įvairiose programose (originaluose) vartojami įvairūs tų pačių ar panašių komandų pavadinimai, pavyzdžiui, komandų grupė *OK*, *Done*, *Finish*, *Apply*, reiškianti atliktų veiksmų patvirtinimą ir įrašymą, ir *Cancel*, *Exit*, reiškianti atliktų veiksmų atsisakymą, išėjimą iš tam tikro lango. Žinant, kuri vidinė funkcija ar vidinių funkcijų seka realizuoja komandą, lokalizacijoje galima įvesti vieningus terminus tą patį veiksmą atliekančioms komandoms, kurios anglų kalba vadinamos skirtingai.

Eilutės identifikatorius (vardas) programos lokalizuojamųjų išteklių sistemoje. Jei eilutėje vartojamas parametras, o parametro vietoje gali būti įrašyta viena iš keleto statinių eilučių iš išteklių, tai šiame atribute gali būti kaupiamas visų tokių eilučių identifikatorių sąrašas. Jei parametro reikšmė gaunama dinamiškai, tai sąrašas tuščias. Jei vietoje parametro įrašoma viena fiksuota eilutė iš išteklių sistemos – sąrašas turi vieną identifikatorių. Atributas taip pat gali būti naudojamas tos pačios komandos tekstą formuojančių suduriamų eilučių identifikatoriams kaupti.

Valdiklio matmenys. Labai svarbus programinės įrangos lokalizavimo aspektas, kuris gali mažinti lokalizacijos kokybę, tai vietos, skirtos tekstams rodyti ekrane, ribojimai.

Paprastai ribojama grafinės sąsajos valdiklio pločiu ir aukščiu. Kai kuriose programose valdiklių plotis yra fiksuotas, kitose – automatiškai prasiplečiantis, prisitaikantis prie išverstos eilutės ilgio. Tačiau ne visos programos „moka“ prisitaikyti, be to, vietos praplėtimas turi ribas. Tyrimai ir lokalizavimo patirtis rodo, kad lokalizuotos eilutės pailgėja daugumoje kalbų. IBM įvertino ir apibendrino teksto pailgėjimo tendencijas verčiant programinės įrangos naudotojo sąsajos tekstus iš anglų kalbos į įvairias Europos kalbas (IBM, 1994) (32 lentelė).

32 lentelė. Vertimų į Europos kalbas ilgiai

Ženklų sk. tekste anglų kalba	Vidutinis vertimo ilgio didėjimas
Iki 10	200–300 proc.
11–20	180–200 proc.
21–30	160–180 proc.
31–50	140–160 proc.
Daugiau kaip 50	130–140 proc.

Kuo trumpesnė sąsajos eilutė, tuo labiau jos ilgis skiriasi įvairiose kalbose. Taigi tikslinga įvesti atributą, nusakantį plotį ir (arba) aukštį, skirtą tekstui rodyti atitinkamame valdiklyje. Tokie atributai gali padėti ne tik matyti pločio (aukščio) ribojimus, bet ir automatiškai skaičiuoti išverstos eilutės ilgį ir pranešti apie vietos trūkumą, jei buvo parinktas per ilgas vertimas. Plotį ir aukštį patogu matuoti tam programinėje įrangoje priimtais matavimo vienetais (pvz., tam tikro ženklo pločiu ar aukščiu, pikseliais ar eilutės teksto ženklais.). Tam tikrais atvejais teksto plotis nėra žinomas, pvz., kai tekstas vietoje parametro įrašomas dinamiškai, programai veikiant. Tuomet skaičiuojant, ar netrūksta vietos, naudojama apytikslė reikšmė.

Sudurtinės eilutės rodiklis. Mūsų parinktoje grafinės sąsajos atvaizdavimo gramatikoje sudurtinės (eilutės, kurios rodomos kompiuterio ekrane greta ir formuoja vieną komandą, pranešimą, užrašą ir pan.) ir alternatyvios eilutės (kai viename valdiklyje vienu metu rodoma viena iš alternatyvių eilučių) aprašomos vienodai. Todėl verta įvesti atributą, kuris nusakytų, ar tam tikro grafinės sąsajos elemento tekstas yra suduriamas iš kelių eilučių, ar ne. Šis atributas taip pat gali būti ir pagalbinis komandos teksto užimamam pločiui skaičiuoti.

Duomenų tipas. Kai išteklių eilutėje pavartotas parametras, kurio vietoje programos vykdymo metu įrašoma tam tikra reikšmė (lokalizavimo metu ji nėra matoma), tai lokalizuotojui naudinga žinoti, koks tos reikšmės duomenų tipas: teksto eilutė, asmens vardas, natūralusis skaičius, dešimtainė trupmena, data, laikas, data ir laikas kartu. Jei parametro reikšmė yra dešimtainė trupmena, tai reikia atkreipti dėmesį į jos skirtuką (taškas ar kablelis), jei data arba laikas, tai reikėtų atkreipti dėmesį į laiko ir datos formatus, laiko ir datos komponentų skirtukus. Jei duomenų tipas skaitinis, tai reikėtų derinti šalia esančio daiktavardžio formą (pvz., 1 objektas, 2 objektai, 100 objektų). Jei yra tokių atvejų, programos autoriai turi pasirūpinti formų derinimo mechanizmo įtraukimu. Jei vietoje parametro įrašomas asmens vardas, tai reikėtų derinti šalia esančio dalyvio ar pusdalyvio giminę (pvz., šiuo metu *Jonienė* yra *prisijungusi*, šiuo metu *Jonas* yra *atsijungęs*). Tam, kad lokalizuotojas įrašytų teisingą reikšmę, tam turi būti paruoštos skirtingos eilutės (pvz., du egzemplioriai eilutės *online*, du egzemplioriai eilutės *offline* iš anksčiau minėto pavyzdžio). Tačiau norint parinkti tinkamą formą naudotojo registracijos metu turi būti numatytas giminę nurodantis laukas, o pagal jo reikšmę automatiškai parenkama tinkamos giminės lokalizuota eilutė, skirta įrašyti prie asmens vardo.

Valdiklio tipas. Teksto eilutės vertimas gali priklausyti nuo valdiklio tipo, kuriame jis bus rodomas, pavyzdžiui, mygtuko pavadinimas dažnai būna veiksmažodinis (*enter new password – įvesti naują slaptažodį*), teksto užrašas (etiketė) turėtų

jau kitokią formą (*enter new password – įveskite naują slaptažodį*), lango pavadinimas turėtų daiktavardinę formą (*enter new password – naujo slaptažodžio įvedimas*). Be abejo, valdiklio tipus ir jų hierarchiją teikia formalios gramatikos išvedimo medžio mazgų pavadinimai ir struktūra (grafinės sąsajos gramatikos dalis). Tačiau jie turi sintaksinę paskirtį, o semantiką perteikia atributai. Todėl verta įdėti valdiklio tipą kaip visos eilutės atributą.

Dialogo lango iškvietimo rodiklis. Žinojimas, ar tam tikra komanda, pavadinta atitinkama eilute iš lokalizuojamųjų išteklių, iškviečia dialogo langą ar ne, gali padėti tiksliau išversti eilutę, parinkti tinkamą formą (dažniau – veiksmažodį), sistemingai laikytis vienodumo visoje programoje, kaip tokias komandas žymėti (pvz., daugelyje programų po tokias komandas įvardijančio teksto rašomas daugtaškio ženklas). Dialogo lango iškvietimo rodiklis taip pat galėtų būti ženklas, kad reikia kartu nagrinėti ir kai kurias to dialogo lango eilutes, lyginti terminiją.

Prieigos ir komandų klavišų tikrinimas. Prieigos ir komandų klavišų temą aptarėme ankstesniame skyrelyje. Lokalizavimo metu tokių klavišų parinkimas kelia nemažai problemų. Visų pirma, reikia parinkti prasmingas raides (klavišus). Antra, jie neturi dubliuotis toje pačioje galiojimo srityje. Specialių atributų, skirtų kaupti lokalizuotų prieigos ir komandų klavišų sąrašui, įvedimas padėtų iš karto aptikti tokių klavišų lokalizavimo klaidas, tikrinti, ar nesidubliuoja tos pačios galiojimo srities lokalizavimo metu parinktas klavišas.

Priskyrus gramatikos simboliams atributus, apibrėžiamos jų semantikos taisyklės – funkcijos, priklausančios nuo kitų atributų reikšmių arba nuo reikšmių, pateikiamų iš išorės. Kai atributinės gramatikos taikomos programavimo kalbos sintaksei ir semantikai formaliai aprašyti, tai atributai skaičiuojami automatiškai pagal pradinių reikšmių aibę (kadangi tos funkcijos nurodo taisykles kompiliatoriui).

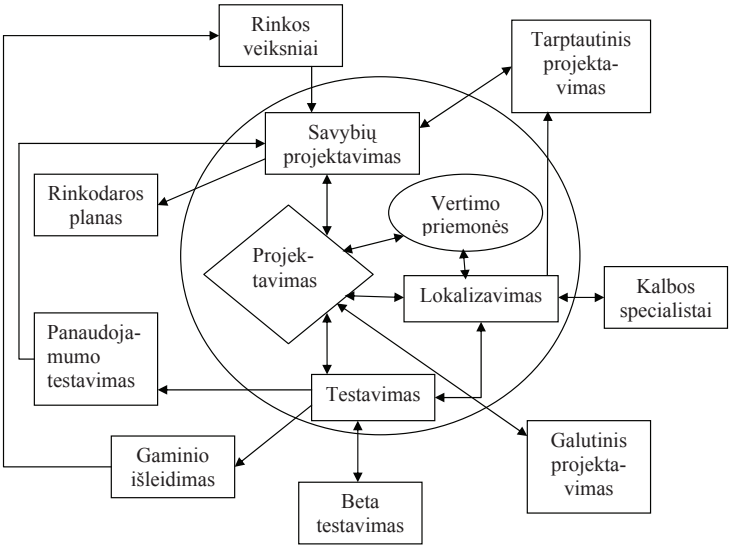
Dirbant su lokalizuojamaisiais ištekliais, visiško automatizavimo siekti nėra tikslinga. Dalis semantikos taisyklių – tai atributo reikšmės priskyrimas iš išorės (pvz., juos priskiria programuotojas, internacionalizavimo ar lokalizavimo inžinierius). Atributams apdoroti taip pat naudojamos papildomos išorinės funkcijos. Parinkus atributus ir aprašius jų semantikos funkcijas, patikrinama, ar atributų priklausomybės grafe nėra ciklų, t. y. ar atributo reikšmės rezultato skaičiavimas nepriklauso nuo pačios to atributo reikšmės.

Čia trumpai apžvelgtos atributinės gramatikos – tai tik vienas žingsnis konteksto formalizavimo link. Galimi ir kiti metodai. Kuris iš jų bus pasirinktas, toliau tobulinamas ir taikomas praktikoje, parodys ateitis.

6. LOKALIZAVIMO PROCESAS

Šiame skyriuje nagrinėjama lokalizavimo darbų struktūra ir jų eiga – lokalizavimo procesas, išskiriami pagrindiniai programinės įrangos kultūros elementai. Nagrinėjant lokalizavimo proceso etapus ir klasifikuojant kultūros elementus apibendrinama programinės įrangos lokalizavimo patirtis, susiję moksliniai šaltiniai ir standartai.

Nadine Kano (1995) pateikia lokalizavimo proceso modelį remdamasi „Microsoft“ bendrovės lokalizavimo patirtimi (47 pav.). Į modelį įtrauktas programinės įrangos projektavimo procesas bei nemažai rinkos veiksnių, kurių nenagrinėsime. Tačiau jis padeda pamatyti projektavimo ir lokalizavimo procesą rinkos kontekste. Jos modelyje laikoma, kad programą projektuoja ir lokalizuoja ta pati bendrovė. Toliau nagrinėsime lokalizavimo procesą, minėtoje schemoje atitinkantį lokalizavimo,



47 pav. Lokalizavimo procesas projektavimo ir rinkos kontekste. Adaptuota remiantis (Kano, 1995)

testavimo ir iš dalies projektavimo komponentus, iš lokalizuotojų (kurie nebūtinai priklauso kūrėjų bendrovei) pozicijos, išskyrę jo svarbiausius aspektus.

Vienareikšmiškai nuspręsti, kuri iš 1-ame skyriuje minėtų dviejų lokalizavimo darbų dalių – programos adaptavimas lokalės normoms ar dialogo tekstų adaptavimas ir vertimas – kelia daugiau problemų, būtų sunku.

Pirmoji darbų dalis – programos adaptavimas – atliekama remiantis formaliaisiais lokalių aprašais.

Antroji – dialogo tekstų vertimas ir adaptavimas – susijusi su didelio kiekio tekstų vertimu ir adaptavimu, jo atnaujinimu, susiduriama su kalbos semantiniais aspektais, kurie nėra formalizuoti nei viename iš egzistuojančių lokalių modelių.

Abiejų lokalizavimo dalių darbai gali būti atliekami lygiagrečiai, tačiau prieš pradedant lokalizuoti svarbu atlikti parengiamuosius darbus. Yang (2007) ir kiti autoriai akcentuoja lokalizuotojų, naudotojų tyrimo grupės ir programuotojų bendradarbiavimą. Net jeigu programinė įranga buvo projektuojama turint omenyje būsimas lokalizacijas (t. y. turi aukštą internacionalizacijos lygį), vis tiek atsiranda problemų, kurias tenka spręsti lokalizavimo metu (Collins, 2002). Tai rodo, kad yra neišspręstų internacionalizavimo problemų ir reikia gilesnės jų analizės.

Iš 47 paveikslo matyti, kad lokalizavimo darbai glaudžiai susiję su testavimu, bendradarbiavimu su kalbininkais. Jei programinę įrangą lokalizuoja ne jos kūrėjai, tai ypač svarbus parengiamųjų lokalizavimo darbų etapas.

6.1. Programos adaptavimas

Lokalizuojamos programos adaptavimas yra pradedamas nuo parengiamųjų lokalizavimo darbų, kurių tikslas – lokalizavimui parinkti labiausiai reikalingas ir pačias geriausias programas, parengti tokį lokalizavimo planą, kad lokalizuotos programos kokybė būtų kuo aukštesnė. Išskirsime pagrindinius veiksmus, kurie atliekami prieš pradedant realius lokalizavimo darbus.

1. **Būsimų naudotojų skaičiaus prognozė**. Prieš pradedant lokalizuoti programą reikia ištirti jos būsimųjų naudotojų skaičių. Laikoma, kad verta imtis programos lietuvinimo, jei būsimų naudotojų skaičius didesnis negu tūkstantis (Grigas, Zalatorius, 2000).

2. **Susipažinimas su programos licencija**. Lokalizavimas yra vienas iš programinės įrangos modifikavimo būdų. Programos licencija – tai teisinis dokumentas, kurio reikia griežtai laikytis. Nuosavybines (ypač komercines) programas paprastai galima modifikuoti, taigi ir lokalizuoti, tik autoriui leidus. Kartais lokalizuoja pats autorius. Nuo licencijos priklauso ir problemų, aptiktų lokalizavimo metu, sprendimo būdas. Jei neleidžiama programą laisvai modifikuoti, problemų sprendimas (internacionalizacijos klaidų šalinimas) priklausys tik nuo autoriaus veiksmų. Esant leidimui modifikuoti pirminius programos tekstus, dalį klaidų gali pašalinti ir lokalizuotojas.
3. **Susipažinimas su informacija apie autorių ir jo darbus**. Iš autoriaus darbų galima spręsti apie jo kvalifikaciją, aktyvumą ir pasirengimą taisyti lokalizavimo metu pastebėtas internacionalizacijos klaidas. Autoriaus pasirengimas aktyviai reaguoti į lokalizuotojo pastabas ypač svarbus, jei programos licencija neleidžia laisvai modifikuoti programos pirminių tekstų.
4. **Susipažinimas su programa**. Lokalizuotojas privalo gerai žinoti lokalizuojamos programos funkcijas, joje vartojamą terminiją. Susipažinti su programos funkcinėmis savybėmis galima tiesiogiai arba iš dokumentacijos. Apie programos kokybę galima spręsti iš recenzijų ir atsiliepimų.
5. **Lokalizavimo galimybių analizė**. Programos išteklių tipai, lokalizuojamųjų išteklių tipai ir jų analizė, programos perkompiliavimo reikalingumas ir pan. Lokalizuojamųjų išteklių tipai išsamiau nagrinėjami 5 skyriuje.
6. **Savo galimybių įvertinimas**. Prieš pradedant lokalizuoti reikėtų įvertinti, ar pakaks jėgų, ištvermės ir išteklių darbui baigti iki galo, nepamiršti, kad atsiras naujų programos versijų ir reikės operatyviai jas lokalizuoti, kad lokalizuotos programos naudotojams kils klausimų ir juos teks konsultuoti.
7. **Kontakto užmezgimas su autoriumi**. Naudingas net ir tuo atveju, kai programos išteklius galima laisvai modifikuoti. Atvirųjų programų lokalizavimo atveju reikėtų užsiregistruoti tos programos tarptautinėje lokalizuotojų bendruomenėje (jei ji yra).
8. **Internacionalizacijos analizė**. Nuo programos internacionalizacijos lygio priklauso lokalizavimo sudėtingumas. Programą lengviau lokalizuoti, jeigu ją projektuojant buvo atsižvelgta į būsimo lokalizavimo poreikius: laikomasi tarptautinių standartų, atskiriami lokalizuojamieji ištekliai (ekrane matomi tekstai, laukų matmenys, koduotės, matavimo vienetai, datų formatai ir pan.) nuo tų, kurie lokalizuojant neturėtų būti keičiami (vykdomieji

Kuo trumpesnė sąsajos eilutė, tuo labiau jos ilgis skiriasi įvairiose kalbose. Taigi tikslinga įvesti atributą, nusakantį plotį ir (arba) aukštį, skirtą tekstui rodyti atitinkamame valdiklyje. Tokie atributai gali padėti ne tik matyti pločio (aukščio) ribojimus, bet ir automatiškai skaičiuoti išverstos eilutės ilgį ir pranešti apie vietos trūkumą, jei buvo parinktas per ilgas vertimas. Plotį ir aukštį patogu matuoti tam programinėje įrangoje priimtais matavimo vienetais (pvz., tam tikro ženklo pločiu ar aukščiu, pikseliais ar eilutės teksto ženklais.). Tam tikrais atvejais teksto plotis nėra žinomas, pvz., kai tekstas vietoje parametro įrašomas dinamiškai, programai veikiant. Tuomet skaičiuojant, ar netrūksta vietos, naudojama apytikslė reikšmė.

Sudurtinės eilutės rodiklis. Mūsų parinktoje grafinės sąsajos atvaizdavimo gramatikoje sudurtinės (eilutės, kurios rodomos kompiuterio ekrane greta ir formuoja vieną komandą, pranešimą, užrašą ir pan.) ir alternatyvios eilutės (kai viename valdiklyje vienu metu rodoma viena iš alternatyvių eilučių) aprašomos vienodai. Todėl verta įvesti atributą, kuris nusakytų, ar tam tikro grafinės sąsajos elemento tekstas yra suduriamas iš kelių eilučių, ar ne. Šis atributas taip pat gali būti ir pagalbinis komandos teksto užimamam pločiui skaičiuoti.

Duomenų tipas. Kai išteklių eilutėje pavartotas parametras, kurio vietoje programos vykdymo metu įrašoma tam tikra reikšmė (lokalizavimo metu ji nėra matoma), tai lokalizuotojui naudinga žinoti, koks tos reikšmės duomenų tipas: teksto eilutė, asmens vardas, natūralusis skaičius, dešimtainė trupmena, data, laikas, data ir laikas kartu. Jei parametro reikšmė yra dešimtainė trupmena, tai reikia atkreipti dėmesį į jos skirtuką (taškas ar kablelis), jei data arba laikas, tai reikėtų atkreipti dėmesį į laiko ir datos formatus, laiko ir datos komponentų skirtukus. Jei duomenų tipas skaitinis, tai reikėtų derinti šalia esančio daiktavardžio formą (pvz., 1 objektas, 2 objektai, 100 objektų). Jei yra tokių atvejų, programos autoriai turi pasirūpinti formų derinimo mechanizmo įtraukimu. Jei vietoje parametro įrašomas asmens vardas, tai reikėtų derinti šalia esančio dalyvio ar pusdalyvio giminę (pvz., šiuo metu *Jonienė* yra *prisijungusi*, šiuo metu *Jonas* yra *atsijungęs*). Tam, kad lokalizuotojas įrašytų teisingą reikšmę, tam turi būti paruoštos skirtingos eilutės (pvz., du egzemplioriai eilutės *online*, du egzemplioriai eilutės *offline* iš anksčiau minėto pavyzdžio). Tačiau norint parinkti tinkamą formą naudotojo registracijos metu turi būti numatytas giminę nurodantis laukas, o pagal jo reikšmę automatiškai parenkama tinkamos giminės lokalizuota eilutė, skirta įrašyti prie asmens vardo.

Valdiklio tipas. Teksto eilutės vertimas gali priklausyti nuo valdiklio tipo, kuriame jis bus rodomas, pavyzdžiui, mygtuko pavadinimas dažnai būna veiksmažodinis (*enter new password – įvesti naują slaptažodį*), teksto užrašas (etiketė) turėtų

6.2. Dialogo tekstų lokalizavimas

Daugelis lokalizavimo bendrovių ir atviruųjų projektų vykdytojų dialogo tekstų vertimo ir adaptavimo išbaigtumą matuoja išverstų teksto eilučių procentu. Tačiau toks matavimas neatspindi tikrovės. Susidūrę su lokalizavimu žino, kad paskutinės paliktos neišverstos (neadaptuotos) lokalizuotinos eilutės sunkiau suprantamos, sunkiau aptinkamos grafinėje sąsajoje. Jose gali būti terminų, dar neturinčių atitikmenų lokalizavimo kalboje. Todėl reikia gerokai daugiau darbo sąnaudų norint jas tinkamai lokalizuoti, nes lokalizuotojas pirmiausia verčia tuos tekstus, kurie jam aiškūs (Esselink, 2000).

Tyrime (Dagienė, Grigas, 2006) pasiūlytas modelis, kuriuo matuojamas lokalizavimo užbaigtumas pagal išverstų eilučių procentą. 33 lentelėje pateikta išverstų eilučių procento ir padaryto lokalizavimo darbo priklausomybė. Išverstų eilučių procentai nurodyti kairiajame stulpelyje ir viršutinėje eilutėje. Pagal šį modelį, pavyzdžiui, jei konstatuojama, kad yra išversta 90 proc. programos sąsajos eilučių, tai reiškia, kad padaryta tik 42 proc. lokalizavimo darbo, jei išversta 96 proc. programos sąsajos eilučių, tai padaryta 58 proc. lokalizavimo darbo.

Dialogo tekstų vertimas ir adaptavimas yra ilgiausias lokalizavimo darbų etapas, kurį galima suskirstyti į smulkesnes dalis (etapus):

1. Juodraštinis dialogo frazių vertimas.
2. Frazių derinimas.
3. Žinyno vertimas ir testavimas.
4. Visų tekstų redagavimas.
5. Kompleksinis testavimas.

33 lentelė. Programos išverstų eilučių procento ir atlikto lokalizavimo darbo procento santykis

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
00	0	0	0	0	0	0	0	0	0	0
10	0	1	1	1	1	1	1	1	1	1
20	1	2	2	2	2	2	2	2	3	3
30	3	3	3	3	4	4	4	4	4	5
40	5	5	5	6	6	6	6	7	7	7
50	8	8	8	9	9	9	10	10	10	11
60	11	12	12	13	13	14	14	15	16	17
70	18	18	19	20	21	21	22	23	24	25
80	26	27	28	29	31	32	34	35	37	39
90	42	43	46	49	51	54	58	62	68	78

Kompiuterio naudotojo matomus tekstus galima suskirstyti į dvi grupes:

1. Sąsajos tekstai:
   - diegimo programos (diegimo vediklio) tekstai;
   - pagrindinės programos tekstai.
2. Ištisiniai tekstai:
   - žinynai, pagalbos tekstai;
   - licencija;
   - kiti tekstai.

6.2.1. Sąsajos tekstai

Tai meniu punktų ir komandų pavadinimai, užrašai ant mygtukų, mygtukų etiketės, kompiuterio pranešimai ir visi kiti tekstai, veikiant programai matomi kompiuterio ekrane. Dauguma jų trumpi, lakoniški, sudaryti iš vieno arba keleto žodžių, išdėstomi dialogo lango plokštumoje (ne tiesiškai). Veikiant programai jų vieta lange gali keistis, tas pats tekstas gali atsidurti įvairiuose languose, greta įvairių kitų tekstų. Ištekliuose paprastai jie pateikiami atskiromis, tiesiškai išdėstytomis eilutėmis. Todėl prarandamas kontekstas. Tik nedaugelis lokalizavimo programų gali juos pateikti tokiu pat pavidalu, kaip jie būna išdėstyti dialogo lange.

Vertėjui, nematančiam konteksto, sunku tekstus teisingai išversti. Ištekliuose greta tikrų eilučių, kurias reikia išversti, paprastai būna galimybė įterpti komentarus. Juose gali būti paaiškinta, kokiame kontekste bus matomas tekstas. Bet komentaras lieka komentaru – jis neformalus, neprivalomas ir nedažnai parašomas. Situacija pagerėtų vietoj komentarų reikalaujant įrašyti formalią informaciją apie kontekstą (5.3 ir 5.4 skyr.). Deja, kol kas tenka pasikliauti komentarais (jeigu jie yra).

Programos sąsajos frazių vertimas skiriasi nuo įprasto rišlaus teksto (straipsnio, knygos ir pan.) vertimo tuo, kad programose vartojamos frazės yra lakoniškos ir atitrūkusios nuo programos veiksmų konteksto, į kurį galima patekti tik dirbant su programa.

Trūkstant konteksto sunku išvengti klaidų. Reikalinga išsami vertimo peržiūra, prilygstanti testavimui ir koregavimui. Tai imlus darbas, kurio apimtis 2–4 kartus didesnė, negu pirminis vertimas. Sudėtingesnis ir redagavimas, nes daugelį redakcinių pataisų reikia priderinti prie konteksto stebint veikiančią programą. Kartais tada išryškėja ir papildomos redaguotinos vietos.

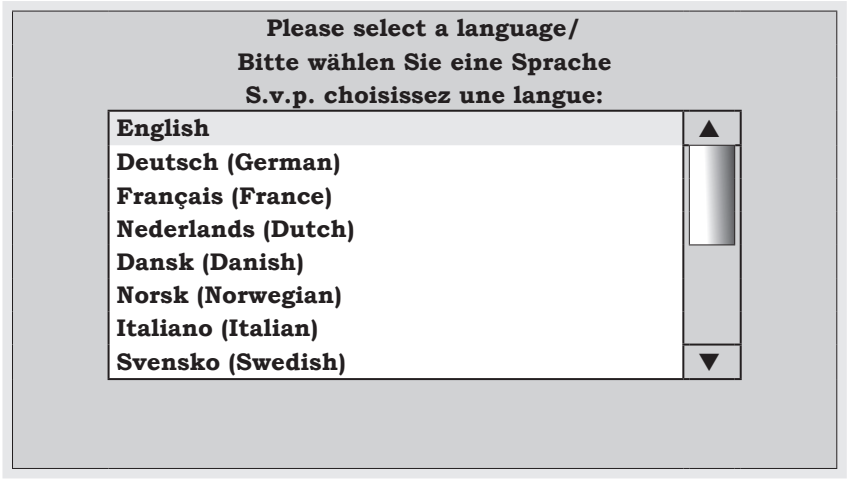
Visa tai, kas pasakyta, tinka ir diegimo, ir pagrindinės programos dialogo langų tekstams. Skiriasi tik tuo, kad jų lokalizuojamieji ištekliai yra atskiruose failuose ir tie gali turėti skirtingą struktūrą.

Diegimo programa panaudojama tik tą kartą, kai diegiama programa, o pagrindinė programa naudojama nuolat. Todėl diegimo programos tekstų panaudojamumas daug kartų mažesnis negu pagrindinės. Dėl to kartais, kai pagrindinės programos naudotojų nedaug, diegimo programa paliekama nelokalizuota. Tačiau tai didelis lokalizuotos programos trūkumas.

Kartais platinamasis programos paketas būna daugiakalbis – į jį sudėti daugelio kalbų lokalizuoti ištekliai, bet pati programa būna vienkalbė – įdiegiama tik vienos, diegimo metu pasirinktos, kalbos ištekliai. Tada iškyla klausimas, kuria kalba programa turi bendrauti su žmogumi programos diegimo metu.

Problemos sprendimo pavyzdys pateiktas 48 paveiksle. Pateikto lango viršuje matome informaciją (pateiktą anglų, vokiečių ir prancūzų kalbomis), ką daryti su žemiau pateiktu sąrašu. Kompiuterio šeimininkas, diegdamas programą, tikriausiai supras bent vieną iš trijų Europoje daugiausia vartojamų kalbų. O jeigu ir nesupras, tai iš sąrašo suvoks, kad reikia pasirinkti kalbą, ir tame sąraše ras savo kalbos pavadinimą, pateiktą ta kalba. Ją pasirinkus, tolesnis dialogas jau gali ir turi vykti pasirinktąja kalba.

Lieka neišverstos tik trys pirmosios skirtingomis kalbomis parašytos eilutės. Lokalizuojant programą nebūtina (dažniausiai – nėra ir galimybės) paaiškinimą papildyti,



48 pav. Kalbos pasirinkimas programos diegimo lange

tarkim, lietuviška eilute. Pakanka ir trijų kalbų, juolab, kad šio užrašo gali ir nebūti – matant kalbų sąrašą ir radus jame savo kalbą, nesunku susivokti, ką toliau daryti.

Daugelis programų autorių daugiakalbiškumo problemą išsprendžia iki galo: pati diegimo programa nustato, kuri kalba pasirinkta operacinėje sistemoje ir iš karto ja „kalbina“ kompiuterio šeimininką.

Sąsajos tekstai būna įvairių formatų, specialiai jiems pritaikytų formatų failuose (žr. 5.2 skyr.).

Visi tekstai (sąsajos frazės, žinynas ir kt.) yra redaguojami. Redagavimas esti dalykinis ir kalbinis. Jei lokalizavimo darbus atliko informatikos specialistai, tai kalbinio redagavimo darbą atlieka redaktorius (lituanistas), turintis kompiuterinių tekstų redagavimo patirtį. Kai programas verčia lituanistai, tai tada situacija turėtų būti priešinga – nereikėtų kalbinio redagavimo, bet reikėtų dalykinio.

6.2.2. Ištisiniai tekstai

Tai ilgi rišlūs tekstai, besitęsiantys per daugelį langų. Dėl to jų vertimas panašesnis į popierinės dokumentacijos vertimą. Todėl galima pasinaudoti taisyklėmis, metodais ir priemonėmis, skirtomis technikos kalbos vertimui. Tačiau yra ir specifikos, kurią ir aptarsime.

Žinynai, pagalbos tekstai. Tai didžiausia tekstų dalis. Kartais vadinama programos dokumentacija. Dėl didelės vertimo apimties žinynai kartais paliekami neišversti, motyvuojant, kad mažai kas juos skaito. Jeigu taip būtų iš tikrųjų, jų nerašytų ir programų originalams.

Programinės įrangos dokumentacijos skaitytojai pagal vertimo poreikį klasifikuojami į tris grupes: 1) profesionalai (programinės įrangos kūrėjai, diegėjai, konsultantai), 2) patyrę naudotojai ir 3) eiliniai naudotojai (Scattergard, 2002). Pirmajai grupei skirtos dokumentacijos nereikia versti (jos apie 25 proc.), antrajai (apie 25 proc.) – dalį versti priklausomai nuo rinkos poreikio, trečiajai (apie 50 proc.) – viską versti.

Pastebėta, kad lokalizuojamų nuosavybinių programų žinynai beveik visada išverčiami, o atvirųjų programų žinynai dažnai lieka neišversti, nors programa laikoma lokalizuota. Tai galima paaiškinti tuo, kad nuosavybinių programų autoriai yra suinteresuoti savos produkcijos pardavimu ir todėl stengiasi kuo geriu patenkinti jų naudotojų poreikius.

Atvirosioms programoms negalioja rinkos dėsniai ir todėl jų lokalizuotojai, kurie patys dažniausiai yra programuotojai, pasirenka lengvesnį kelią.

Daugiakalbėse programose pasitaiko, kad žinynas išverstas ne į visas kalbas, į kurias išversta sąsaja, o nesant jo vertimo siūloma pasirinkti kurios nors kitos kalbos, kurią galbūt pakankamai gerai moka programos naudotojas, žinyną. Tai geriau, negu be pasirinkimo pateikti tik originalo kalbos žinyną.

Beveik visa programinė įranga, verčiama į lietuvių kalbą, skirta eiliniam naudotojui, todėl jų žinynų vertimą laikysime neatskiriama lokalizacijos dalimi.

Žinyno vertimas panašus į įprastų tekstų vertimą, todėl paprastai rimtų lokalizavimo problemų nesukelia. Tačiau verčiant žinyną dažniausiai atliekamas ir programos testavimas – dirbama su veikiančia programa ir tikrinama, ar viskas, kas išversta, teisinga. Tuo pat metu testuojami bei koreguojami ir dialogo frazių vertimai, kadangi verčiant ir nuodugniai nagrinėjant žinyną išryškėja programos taikymo ir veiksmų kombinacijos, kurios nebuvo numatytos. Testavimo darbų dalis yra didesnė už patį vertimą. Galima teigti, kad tik išvertus (arba parašius originalų) žinyną bus užbaigtas ir visas programos lokalizavimas (Scattergard, 2002). Be to, žinyno tekstą reikia adaptuoti (dažniausiai papildyti) turint omenyje, kad programa veiks konkrečioje kalbos ir kultūros terpėje.

Programos žinyno vertimas sudėtingesnis negu aparatūros žinynų, nes reikia jo tekstą suderinti su sąsajos tekstų vertimais: turi būti vartojami tokie patys sąsajos elementų (meniu punktų, mygtukų, parinkčių ir kt.) pavadinimai.

Paveikslai ir kitos iliustracijos turi būti pakeisti tais, kurie iš tikrųjų matomi lokalizuotoje programoje. Taigi reikia išversti ne tik aprašymus, bet ir iliustracijose esančius tekstus.

Programų žinynuose būna spragų – kas nors nepakankamai aprašyta arba išvis neaprašyta. Be to, lokalizuojant atsiranda naujų dalykų, kuriuos naudinga turėti aprašytus. Dėl to dažnai žinyną nepakanka vien tik išversti, bet reikia ir adaptuoti.

Programos žinynas nėra literatūrinis kūrinys, tad nėra labai svarbu, kad vertimas tiksliai atspindėtų jo originalo autoriaus stilių, pažiūras ir pan., bet svarbu, kad tiksliai, išsamiai, suprantamai ir vienareikšmiškai būtų aprašytas programos veikimas. Todėl būtinas kompleksinis testavimas, t. y. žinyno vertimo testavimas su veikiančia programa, kartu testuojant ir visą programą. Tada sutikrinama, ar viskas, kas žinyne parašyta, yra tiesa.

Originalių programų žinynuose būna spragų – kas nors nepakankamai aprašyta arba išvis neaprašyta. Lokalizavimo metu reikia pasistengti jas užpildyti. Be to, lokalizuojant atsiranda naujų dalykų, kuriuos reikia aprašyti žinyne. Dėl visų čia išvardytų priežasčių prie žodžio „vertimas“ dažniausiai reikia pridėti „ir adaptavimas“.

Žinynas gali būti įdėtas į programą arba laikomas internete (saityne), o programoje – tik saitas į jį. Internetinį žinyną galima operatyviau koreguoti, naujinti. Tačiau juo galima naudotis tik prisijungus prie interneto. Dėl to jis dažniau naudojamas tik internetinėse programose, pavyzdžiui, naršyklėse, kurios ir taip būna prijungtos prie interneto.

Žinynuose būna daug vidinių ir išorinių saitų. Saituose esantys adresai neverčiami, nes juos mato tik lokalizuotojas.

Žinynuose, ypač internetiniuose, būna saitų į tinklalapius, kuriuose galima rasti papildomos informacijos, patarimų. Svarbesni tinklalapiai, būtinai reikalingi žinyno skaitytojui, pavyzdžiui, paaiškinantys kurias nors programos dalis ar funkcijas, yra verčiami.

Tinklalapių tekstai nėra originalios programos dalys. Jiems galioja tuose tekstuose nurodytos autoriaus teisės, panaudojimo sąlygos ir kitos taisyklės, kurių lokalizuotojams privalu laikytis.

Jeigu tinklalapyje pateikiama tik nebūtina papildoma medžiaga, tai gali būti neišversta. Tokiu atveju po nuorodos į tinklalapį skliaustuose nurodoma jo kalba (pvz., (*anglų k.*)).

Jeigu papildomų tinklalapių autoriai nėra originalios programos autoriai, tai šiuos tinklalapius versti galima tik laikantis jų naudojimo sąlygų ir autorių teisių reikalavimų, nurodytų tuose tinklalapiuose.

Internetinius žinynus galima laikyti nedidelėmis interneto svetainėmis. Jie pateikiami tokiu pat formatu (pvz., HTML, XHTML), kaip ir kiti tinklalapiai.

Tokiais pačiais formatais pateikiami ir programos viduje esantys žinynai. Tačiau čia naudojami ir kitokie formatai, specialiai skirti programų žinynams. „Windows“ sistemos programose – HLP ir CHM formatai. Tokie žinynai dažniausiai rašomi „Word“ tekstų rengykle. Saitai užrašomi specialiais ženklais puslapių išnašose. Žinyno skaitytojas jų nemato. Todėl su jais elgiamasi taip pat, kaip ir su kitais žinyno saitais – jie neverčiami ir nekoreguojami.

Licencija. Tai teisinis dokumentas – sutartis tarp programos autoriaus ir jos naudotojo. Todėl ji turi būti verčiama tiksliai (neadaptuojama). Vertėjas arba vertimo aprobuotojas turi turėti atitinkamą dokumentais patvirtintą vertėjo kvalifikaciją.

Pagal Lietuvos Respublikos įstatymus sutartis su užsienio fiziniais arba juridiniais asmenimis turi būti sudaroma valstybine arba abiems pusėms priimtina kalba. Dėl to nuosavybinių programų lokalizuotojai visada pasistengia teisiškai kvalifikuotai išversti jų licencijas.

Problemų yra su atviրųjų programų licencijomis. Jos būna parašytos anglų kalba ir jose būna punktas, kuriame sakoma, kad kilus teisminiams ginčams bus vadovaujamasi angliška licencijos versija. Išversti licenciją galima, bet vertimas neturės juridinės galios ir apie tai turi būti parašyta vertimo pradžioje. Susidaro parodoksali situacija. Angliška licencija negalioja dėl to, kad ji parašyta ne valstybine kalba, o lietuviška negalioja dėl to, kad ji parašyta ne anglų kalba.

Bet atviրųjų programų licencijos labai liberalios. Jų licencijuojamas programas galima įdiegti į neribotą kompiuterių skaičių, galima platinti be jokių ribojimų. Taigi eilinis naudotojas netgi neturi galimybės tokią licenciją pažeisti.

Atviրųjų programų licencijose kalbama apie programų modifikavimą, tobulinimą, jų komponentų panaudojimą kitose programose, taigi faktiškai jos skirtos programuotojams ir lokalizuotojams. Tam, kad įstatymus gerbiantis eilinis programos naudotojas negaištų laiko skaitydamas painų juridinį tekstą, nors jis būtų išverstas į lietuvių kalbą, prieš licenciją naudinga aiškiai parašyti, ką jis gali daryti su programa. Pavyzdžiui, pateikti tokį tekstą:

Čia pateikiamas <Licencijos pavadinimas> licencijos neoficialus vertimas į lietuvių kalbą. Ši licencija praktiškai neriboja programos naudojimo darbui ir jos kopijavimo. Tačiau tiems, kas nori programą modifikuoti arba ją ar jos komponentus panaudoti komerciniams tikslams, reikia nuodugniai išsiaiškinti teisinius aspektus ir atidžiai perskaityti po vertimu pateiktą originalų licencijos tekstą anglų kalba, nes kilus teisiniams ginčams bus vadovaujamasi angliškąja licencijos versija.

Kiti tekstai. Tai failas *readme.txt (skaityk.txt)* su pradine informacija apie programą, kompiliatorių programose – programavimo kalbų aprašymai, įvairi su programa susijusi metodinė arba mokomoji medžiaga. Tai rišlūs technikos kalbos tekstai. Jie gali būti parengti bet kuriuo tekstiniams dokumentams tinkamu ir visuotinai naudojamu formatu. Gali būti skirti lokalizuotojams, aptarnaujančiam personalui. Verčiami tie, kurių gali prireikti programos naudotojui.

6.2.3. Ką reikia ir ko nereikia versti, ką galima ir ko negalima versti

Versti reikia viską, ***ką kompiuterio naudotojas mato ekrane***: meniu užrašus, dialogo langus, užrašus ant mygtukų, kompiuterio pranešimus, žinynus, pagalbos failus ir kitus failus su kompiuterio naudotojui skirtais tekstais (pvz., failą *readme*).

Dalis tekstų skirti aukštesnio lygio naudotojui – *sistemos administratoriui.* Kartais abejojama, ar juos reikia versti. Bet daugelis asmeninių kompiuterių savininkų

patys tvarko savo kompiuterius naudodamiesi administratoriaus teisėmis. Jeigu nebus išversti administratoriui skirti tekstai, tai gerai nemokantiems anglų kalbos bus sunkiau tomis teisėmis pasinaudoti ir dažniau teks kreiptis į specialistą. Net ir tuo atveju, kai įstaigoje yra kompiuterius tvarkantis specialistas, atliekantis administratoriaus funkcijas, parankiau turėti išverstus jam skirtus tekstus, kad nekiltų problemų dėl terminijos ir būtų lengviau susikalbėti su paprastais kompiuterių naudotojais.

Būna tekstų, skirtų *programos tobulintojams*. Jų galima neversti, nes tobulintojų nėra daug. Be to, jie turi mokėti kalbą, dominuojančią programinės įrangos gamyboje (šiuo metu anglų) dėl savo darbo specifikos. Kartais į jiems skirtų žinyno skyrių pradžią įterpiama tuose skyriuose esančios medžiagos santrauka, parašoma, kam ta medžiaga skirta.

Dalis tekstų skirti kitakalbiams, kuriems gali tekti susidurti su lokalizuota programa. Lokalizuojamuose ištekliuose būna tekstų, skirtų tik lokalizuotojams. Kai kurių jų nerekomenduojama versti, o kai kurie neturi būti verčiami. Aptarsime abejones keliančias atskiras tekstų grupes.

Kalbų pavadinimai. Kalbos programinėje įrangoje įvardijamos trejopai:

1. Ta pačia kalba, kaip ir sąsajos kalba.
2. Pačios kalbos kalba.
3. Kalbos kodu.

1. Ta pačia kalba, kaip ir sąsajos: originalioje programoje – originalo kalba, lokalizacijoje – lokalizacijos kalba. Taigi, jeigu angliškame originale kalba pavadinta angliškai, reikia jos pavadinimą išversti.

Valstybinės lietuvių kalbos komisijos patvirtintus kalbų pavadinimus galima rasti Vikipedijos tinklalapyje „Kalbų klasifikatorius“[1] arba Lietuvos Respublikos Seimo svetainėje[2]. Ilgesnis kalbų pavadinimų sąrašas pateiktas tinklalapyje http://ims.mii.lt/kalba.

Atkreipiame dėmesį, kad kalbų pavadinimai anglų kalba rašomi iš didžiųjų raidžių, lietuvių kalba ir beveik visomis kitomis kalbomis – iš mažųjų (be abejo, jei ne sakinio pradžioje).

---

[1] Vikipedija. Kalbų klasifikatorius, http://lt.wikipedia.org/wiki/Kalbų_klasifikatorius

[2] Lietuvos Respublikos Seimas. Įsakymas „Dėl kalbų, tranzito dokumentų, Lietuvos Respublikos apskričių klasifikatorių, lydinčiojo dokumento formų A ir B, prekių sąrašo formų A ir B patvirtinimo“. 2003 m. sausio 31 d. Nr. 1B-88. http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=205028&p_query=&p_tr2=

2. Kalbų pavadinamai rašomi jų pačių kalbomis, pavyzdžiui,

Dansk;
Deutsch;
Ελληνικά;
Euskara;
תירבע.

Tokie sąrašai vartojami daugiakalbėse programose arba interneto svetainėse (pvz., Vikipedijoje), kad tų kalbų atstovai galėtų pamatyti jiems gerai pažįstamą savos kalbos pavadinimą ir tą kalbą pasirinkti. Todėl juos reikia palikti originalius.

Dar geriau, kai programos originale būna dvikalbiai kalbų pavadinimai: originalia kalba ir programos originalo kalba, pavyzdžiui,

Dansk (Danish);
Deutsch (German);
Ελληνικά (Greek);
Euskara (Basque);
תירבע (Hebrew).

Tuos, kurie parašyti programos originalo kalba, tikslinga išversti:

Dansk (danų);
Deutsch (vokiečių);
Ελληνικά (graikų);
Euskara (baskų);
תירבע (hebrajų).

3. Kalbų kodai būna dviraidžiai arba triraidžiai. Jie nustatyti tarptautiniu standartu ISO 639. Pirmoje standarto dalyje ISO 639-1 apibrėžti dviraidžiai kodai, antroje – ISO 639-2 – triraidžiai. Triraidžių kodų daugiau, todėl juos turi ir mažiau vartojamos kalbos, kurioms nepakaktų dviraidžių kodų.

Kalbų kodai unikalūs, vienareikšmiškai įvardija kalbas tekstuose bet kuria kalba. Juos galima laikyti tarptautiniais kalbų vardais. Todėl jie patogūs profesionalams – programuotojams, programų lokalizuotojams ir kitiems, turintiems reikalų su dideliu kalbų skaičiumi. Jie neverčiami. Tačiau verta prisiminti, kad paprasti programų naudotojai žino tik nedaugelio kalbų kodus (pvz., LT, LV, EN, bet kažin ar daugelis pasakys, kad ES yra ispanų, o ne estų kalbos kodas, o SV slovėnų, o ne slovakų). Todėl verčiant programą kalbų kodais nereikia per daug žavėtis ir neimti jais keisti kalbų pavadinimų. Atvirkščiai, rišliame tekste (pvz., žinyne) kartais gali būti tikslinga kodą pakeisti įprastu kalbos pavadinimu.

Valstybių pavadinimai. Valstybių pavadinimai, kaip ir kalbų pavadinimai, gali būti rašomi trejopai: programos sąsajos kalba, oficialia valstybės kalba ir valstybės kodu.

Dažniausiai valstybės pavadinimas rašomas kartu su kalba tam, kad būtų galima skirti kalbos dialektus, vartojamus skirtingose valstybėse. Pavyzdžiui, vokiečių kalba Vokietijoje ir vokiečių kalba Šveicarijoje. Tada sakoma (ir rašoma rišliame tekste): *Vokietijos vokiečių, Šveicarijos vokiečių* ir t. t. Angliškuose tekstuose valstybė dažniau rašoma skliaustuose po kalbos pavadinimo. Todėl, pavyzdžiui, sakinį *The last paragraph is translated into Germam (Swiss)* reikėtų versti: *Pastaroji pastraipa išversta į Šveicarijos vokiečių kalbą*. Tačiau sąrašuose, iš kurių kalba pasirenkama, racionaliau pirmiau rašyti kalbą, o po jos, paprastai skliaustuose, – valstybę (arba kitą teritoriją, kurioje vartojamas tos kalbos dialektas). Tokiame pagal abėcėlę sutvarkytame sąraše visi tos pačios kalbos variantai atsiduria greta:

Ispanų (Ispanija);
Ispanų (Lotynų Amerika);
Prancūzų (Kvebekas);
Prancūzų (Prancūzija);
Vokiečių (Austrija);
Vokiečių (Šveicarija);
Vokiečių (Vokietija).

Valstybių ir teritorijų kodai nustatyti ISO 3166-1 standartu. Jame apibrėžti dviraidžiai, triraidžiai ir skaitiniai (trijų skaitmenų kodai) valstybių ir teritorijų kodai. pavyzdžiui:

Lietuvos kodai – LT, LTU ir 440;
Latvijos – LV, LVA, 428;
Estijos – EE, EST, 233.

Programose dažniausiai vartojami dviraidžiai kodai. Kalba, kuria kalbama kurioje nors valstybėje, įprasta užrašyti kalbos kodu (mažosiomis raidėmis) ir brūkšneliu atskirtu valstybės kodu (didžiosiomis raidėmis), pavyzdžiui, Austrijos vokiečių: „de-AT“.

Programų vardai. Iš lokalizavimo praktikos galima pastebėti, kad dažniau verčiami arba adaptuojami vietinės reikšmės programų vardai, pavyzdžiui, *Comenius Logo → Komenskio Logo, Geometer's Sketchpad → Dinaminė geometrija*. Visuotinai naudojamų programų vardai (pvz., *Windows, Word, Mozilla*) neverčiami. Tačiau į jų sudėtį įeinančių programų vardai verčiami, pavyzdžiui, „Windows“ sistemoje: *Note-*

pad → Užrašinė, Wordpad → Tekstų doroklis, Calculator → Skaičiuotuvas. „Mozillos“ (dabar „SeaMonkey“) programų pakete: *Navigator → Naršyklė, Mail & Newsgroups → Paštas, Composer → Rašyklė*.

Daugelio programų vardai turi prekės ženklo statusą. Ar galima tokį vardą versti, transliteruoti, transkribuoti, nustatoma to ženklo registracijos dokumentuose. Tai natūrali autoriaus teisė į jo sukurto produkto vardą, netgi jeigu vardas nėra registruotas prekės ženklas. Bet kuriuo atveju dėl oficialaus programos vardo vertimo lokalizuotojas turi tartis su originalo autoriumi, kuris yra ir jo vardo savininkas. Reikia turėti omenyje, jog kiekvienas gamintojas suinteresuotas, kad prekė būtų paklausi, t. y. patiktų pirkėjui. Tai svarus lokalizuotojo, netiesiogiai atstovaujančio būsimo gaminio pirkėjo interesus, argumentas derybose su programos autoriumi.

Skirtumų tarp gamintojo ir naudotojo būtų mažiau, jeigu originalių programų vardai būtų parenkami apgalvotai. Tokie, kad būtų lengvai įsimenami ir ištariami[1]. Tada būtų mažesnis poreikis juos versti. Tokių pavyzdžių yra: pašto programos „Balsa“, „Eudora“, naršyklė „Opera“.

Atvirųjų programų pirminius tekstus gali laisvai platinti, modifikuoti, panaudoti savo kuriamose programose bet kas. Svarbu tik, kad iš jų pagamintos programos būtų atvirosios ir jų tekstuose būtų nurodyti panaudotų tekstų autoriai. Lokalizavimas prilygsta programos modifikavimui. Todėl lokalizaciją galima laikyti nauja programa ir bet kaip ją pavadinti. Tačiau tokiu atveju visi darbai, susiję su programos kompiliavimu, klaidų taisymu, priežiūra ir platinimu atitektų lokalizuotojui. Operatyviau ir ekonomiškiau juos būtų daryti centralizuotai, visų kalbų lokalizacijoms iš karto. Todėl juos paprastai atlieka originalo autorius. Dėl to autorius programos vardą dažnai laiko prekės ženklu ir tada reikia laikytis to ženklo naudojimo taisyklių.

Prie programos vardo būna aiškinamųjų žodžių, pavyzdžiui, „Home Edition“, „Enterprise Edition“, „Lite“, „Lite Version“. Tai ne simboliniai vardai, o normalūs žodžiai, kurie čia vartojami jiems įprasta prasme ir turėtų būti suprantami programos naudotojui. Jie nėra ir negali būti prekės ženklai, todėl jie turėtų būti verčiami.

Kartais programos autoriai pataria ir kaip tai padaryti. Pavyzdžiui, „Mozillos“ prekės ženklų naudojimo taisyklėse[2] atskirų kalbų bendrijų savarankiškai išleidžiamas naršyklės „Firefox“ versijas siūlo vadinti „Firefox Community Edition“ (čia „Fire-

[1] A. L. Brown. Introduction to Marketing, University of Delaware, 1996. http://www.udel.edu/alex/sylla.html, http://www.udel.edu/alex/chapt12.html#selecting

[2] Localization Trademark Policy – Mozilla Firefox. http://www.mozilla.org/foundation/trademarks/l10n-policy.html

fox“ yra prekės ženklas), bet žodžius „Community Edition“ galima išversti ir dar ką nors prie jų pridėti. Pateiksime taisyklėse pateiktus pavyzdžius: „Firefox Community Edition, French“, „Polish Thunderbird Community Edition“, „Firefox Community Edition, by the German Localization Team“. Juose, viskas, išskyrus žodį „Firefox“, gali būti išversta į lokalizacijos kalbą. Taisyklėse sakoma, kad žodžių „Community Edition“ galima ir išvis nevartoti, bet tada pavadinimą reikia suderinti su originalo autoriais (lietuvišką versiją buvome pavadinę „Firefox LT“).

Aplankų ir failų vardai. Interneto svetainėje[1] aptariamas aplankų ir failų vardų vertimas. Aplankų, kuriuos naudoja operacinė sistema, vardų nereikia versti, nes tada sistema jų neras. Tokių aplankų pavyzdžiai šakniniame „Linux“ sistemos aplanke: *home, mnt, usr, etc, bin, sbin*; „Windows“ sistemos šakniniame aplanke: *Windows, Windows\System*. Vardus tų aplankų ar failų, kuriuos naudoja kompiuterio naudotojas, reikia išversti, pavyzdžiui,

readme – skaityk,
My Documents – Dokumentai,
Spam – Brukalas.

Kitų aplankų ir failų vardus, kurių nenaudoja kitos programos ir kurių nemato naudotojas, vertimas nepakenks ir nepadės naudotojui, todėl juos galima versti arba neversti. Failų prievardžiai neverčiami.

Vidinių saitų vardai. Jų būna žinynuose, pagalbos failuose ir kituose programų dokumentacijos failuose. Jie nematomi naudotojui. Netikslumai jų vertimuose nebūtų matomi tiesiogiai, bet sukeltų naršymo klaidų. Todėl jie neverčiami.

Lokalizuojamųjų išteklių eilučių parametrai. Juos mato tik programos lokalizuotojas. Todėl jų nereikia versti (daugiau apie juos žr. 4.2.8 skyr.).

Komandų vardai tekstinėje sąsajoje. Jie įkoduoti į tas komandas atliekančias programas. Todėl versti negalima, nes jie turi būti tokie pat, kaip ir tose programose. Grafinėse sąsajose jie vartojami retai. Tačiau tų pačių komandų vardai, vartojami grafinėje sąsajoje (pvz., užrašyti ant mygtukų), tiesioginio (tekstinio) ryšio su programa neturi ir juos ne tik galima, bet ir reikia išversti.

Komentarai išteklių eilutėse skirti vertėjui. Vertėjas juos perskaito, jais pasinaudoja versdamas juose paaiškintą tekstą ir tuo komentarų tarnyba baigiasi. Taigi nėra tikslo juos versti. Bet jie paliekami ir lokalizuotuose ištekliuose (neišbraukiami) – gal jų prireiks ateityje taisant vertimo klaidas, atnaujinant lokalizaciją ir pan.

[1] Translate Toolkit & Pootle, http://translate.sourceforge.net/wiki/guide/translation/program_names

6.2.4. Vertimo iš anglų kalbos į lietuvių kalbą specifika

Didžiosios raidės. Didžiųjų raidžių vartojimas skirtingose kalbose skiriasi. Pavyzdžiui, vokiečiai visus daiktavardžius rašo iš didžiųjų raidžių. Anglai iš didžiųjų raidžių rašo savaitės dienų, mėnesių, kalbų pavadinimus, visus žodžius (kartais – išskyrus artikelius, prielinksnius ir kitas smulkmenas) antraštėse, įstaigų pavadinimuose. Jeigu angliškas žodis užrašytas ne sakinio pradžioje arba nėra tikrinis daiktavardis, tai beveik visada jo lietuvišką atitikmenį reikia rašyti iš mažosios raidės.

Meniu punktų pavadinimai, užrašai ant mygtukų ir kiti atskirai esantys užrašai laikomi atskirais sakiniais, nors ir būtų sudaryti iš vieno žodžio. Todėl ir lietuviškame vertime juos reikia rašyti iš didžiųjų raidžių.

Atskirais atvejais, kai žemesnio lygio meniu punktų užrašai yra gramatiškai susiję su aukštesnio lygio meniu ir laikytini sakinio, kurio pradžia yra aukštesniame punkte, tęsiniu, žemesnio punkto eilutę reikia rašyti iš mažosios raidės, pavyzdžiui:

Eiti į
ankstesnį
tolesnį
pradžią
pabaigą

Žodžių tvarka sakinyje. Anglų kalbos sakinyje žodžių tvarka griežta: veiksnys, tarinys, papildinys, aplinkybės. Lietuvių kalboje žodžių tvarka laisva ir pasinaudojant šia savybe sakiniui galima suteikti ekspresyvumo. Pavyzdžiui:

Only I can see results → Rezultatus galiu matyti tik aš.
You cannot send an empty message → Tuščio pranešimo siųsti negalima.

Skyrybos ženklai. Anglų ir lietuvių kalbose skyrybos ženklų vartojimas ne visada sutampa. Pavyzdžiui, kai išvardijami objektai skiriami kableliais, tai anglai deda kablelį ir prieš paskutinius objektus jungiančius jungtukus *and*, *or*:

Red, green, and blue colors must be used → Turi būti panaudotos raudona, žalia ir mėlyna spalvos;
Red, green, or blue color must be used → Turi būti panaudota raudona, žalia arba mėlyna spalva.

Santrumpos. Anglų kalboje santrumpos vartojamos dažniau negu lietuvių. Nemažai būna ir dirbtinai sudarytų santrumpų, kurių raides atitinkantys žodžiai nelabai informatyvūs, dirbtinai „pritempti“. Jie greitai pamirštami ir lieka funkcionuoti tik santrumpa.

Originaliose programose santrumpų būna daugiau dar ir dėl to, kad jas mėgsta vartoti programuotojai. Jų aplinkoje santrumpos suprantamos ir čia jų vartojimas nekelia problemų. Tačiau kai jos patenka į visiems kompiuterių naudotojams skirtus tekstus, darosi nesuprantamos.

Lokalizuojant programinę įrangą reikia į tai atkreipti dėmesį ir mažinti santrumpų skaičių jas keičiant pilnais sąvokų pavadinimais arba paaiškinti (išskleisti) santrumpas, kai jos panaudojamos pirmą kartą. Būna atvejų, kad santrumpą galima išvis praleisti, kartais ir su visa fraze. Lokalizuotoje programoje turi likti tik visuotinai žinomos santrumpos arba bent paaiškintos.

Verčiant programą į lietuvių kalbą reikia turėti omenyje, kad lietuvių kalba yra sintetinė, o santrumpos nelinksniuojamos, neturi giminės požymio. Todėl mažesnį jų vartojimą lemia ne tik tradicija, bet ir kalbos struktūra.

Asmeniniai ir savybiniai įvardžiai. Anglų kalboje vartojami žymiai dažniau, negu lietuvių. Kai kuriose sakinio vietose jie tiesiog privalomi. Palikus juos ir lietuviškame sakinyje dažniausiai sakinio prasmė išlieka, bet lieka nereikalingų žodžių. Todėl verčiant reikia pagalvoti apie kiekvieną įvardį, ar lietuviškame tekste jis tikrai reikalingas. Pateiksime pavyzdžių:

I *Agree* → *Sutinku.*

I *Disagree* →*Nesutinku.*

Thank ***you*** *for* ***your*** *attention* →*Ačiū už dėmesį.*

Choose ***your*** *recipient* → *Pasirinkite gavėją.*

Check ***my*** *spelling as* ***I*** *type* → *Tikrinti rašybą renkant tekstą.*

Are ***you*** *sure* ***you*** *want to remove this photo* → *Ar tikrai norite pašalinti šią nuotrauką.*

You *cannot send an empty message* → *Tuščio pranešimo siųsti negalima.*

If ***you'd*** *like,* ***you*** *may skip this step* → *Jei norite, šį žingsnį galite praleisti.*

Your *password is incorrect* → *Slaptažodis neteisingas*. (O kieno gi kito slaptažodis gali būti?).

Load photos from ***your*** *mobile phone. Įkelkite nuotraukas iš mobiliojo telefono.* (Įkėlimui nesvarbu, ar telefonas savas)

Kreipdamiesi į vieną asmenį ar daugelį anglai vartoja tą pačią daugiskaitinę įvardžio formą *you*. Dažnai, kaip ir matome pateiktuose pavyzdžiuose, šį įvardį lietuviškame vertime galima praleisti. Tačiau kai kreipiamasi į asmenį tiesiogiai, jis reikalingas. Lietuvių kalboje vartojamos dvi formos: neformali vienaskaitinė *tu* ir oficiali daugiskaitinė *jūs,* kuria reiškiama pagarba asmeniui, į kurį kreipiamasi. Kurią formą vartoti programose, kai programa kreipiasi į jos naudotoją? Derėtų gerbti naudoto-

ją ir vartoti kreipinį *jūs*, o atskirais atvejais, kai tinka familiarumas ir betarpiškumas, pavyzdžiui, vaikiškų žaidimų programose, galima vartoti ir *tu*.

Kreipinyje į vieną asmenį žodis *jūs* paprastai rašomas iš didžiosios raidės (*Jūs*), nes tai geriau tinka pagarbai išreikšti.

Pagalbiniai veiksmažodžiai. Anglų kalbos veiksmažodžiai turi sudėtingą ir tikslią laikų sistemą, kurioje gausiai vartojami pagalbiniai veiksmažodžiai. Verčiant jie, ypač *is* → *yra*, *was* → *buvo*, „prasmunka“ ir į lietuvišką tekstą. Čia dažniausiai jie būna nereikalingi, verčiant juos galima praleisti. Pateiksime pavyzdžių:

This field ***is*** *required* → *Šis laukas privalomas.*

The message ***is*** *too long* → *Pranešimas per ilgas.*

The message ***was*** *successfully sent* → *Pranešimas išsiųstas sėkmingai.*

Kiti nereikalingi žodžiai. Tekstuose būna nereikšmingų žodžių ir frazių, kuriuos išbraukus neprarandama informacijos, nesuprastėja teksto logiškumas ir aiškumas. Juos išbraukus tekstas sutrumpėja ir jam suprasti reikės mažiau pastangų. Pateiksime pavyzdžių iš konkrečių programų:

This field is required ***by the user in order to submit the form*** → *Šis laukas privalomas*

Leave *comments* → *Komentuokite.*

Translation ***submitted*** *misses* ***one or more*** *HTML tags* → *Vertime trūksta HTML gairių.*

Copyright © 2007 Microsoft → *© 2007 Microsoft*

Copyright © by Werner Böme, Translation into German → *© Werner Böme, vertimas į vokiečių kalbą.*

Tą pačią mintį kartais pavyksta išreikšti trumpiau, pavartojus kitus žodžius, pavyzdžiui,

Search for people who share the same last name with you → *Ieškoti bendrapavardžių.*

Papildomi žodžiai. Kartais reikia įterpti papildomų žodžių, kurie anglų kalbos tekste gali būti praleisti (numanomi), bet reikalingi rišliame lietuviškame tekste:

Valid until 2015 → *Galioja iki 2015* ***m.***

Lithuanian grammar → *Lietuvių* ***kalbos*** *gramatika.*

Passwords are case-sensitive → *Slaptažodžiuose didžiosios ir mažosios raidės laikomos skirtingomis.*

Passwords are case-sensitive → *Slaptažodžiuose ABC ir abc (eilutės) laikomos skirtingomis.*

*Edit connection → Taisyti ryšio **aprašą**.*
*Edit developers → Koreguoti projektuotojų **duomenis**.*

Simboliniai vardai ir jų vartojimas. Programose būna nelietuviškos kilmės simbolinių vardų. Tai programų vardai, interneto svetainių pavadinimai ir kt. Patarimuose lokalizuotojams[1] bendru atveju, nenurodant konkrečios kalbos, į lokalizacijų kalbas neišverstus programų vardus siūloma išskirti apostrofais.

Pagal lietuvių kalbos taisykles (Sližienė, Valeckienė, 1992) simboliniai vardai turi būti išskirti kabutėmis arba kitokiomis priemonėmis, pvz., kitu šriftu:

„Windows“ operacinė sistema;
Windows operacinė sistema.

Dažniau vartojamas kursyvas, nes juo išskiriamas žodis paryškinamas tiek, kad būtų išskirtas, bet ne per daug, kad atkreiptų (ir blaškytų) skaitytojo dėmesį.

Galimybės vartoti šriftus sąsajos tekstuose ribotos. Todėl paprastai vartojamos kabutės. Jei yra galimybė, vartojamas kursyvas, kuris dažniausiai nurodomas HTML gairėmis, pavyzdžiui,

Windows operacinė sistema.

Vardų išskirti nereikia, jeigu jie parašyti atskirai, t. y. nerišliame tekste ir nėra iš ko juos išskirti, pavyzdžiui, meniu punkto pavadinime, atskiroje teksto eilutėje, lentelės langelyje ir pan. Tai taikoma ir keliems greta parašytiems vardams, kurie vienas nuo kito gali būti skiriami kableliais ar kitokiais skyrybos ženklais. Taip pat nereikia išskirti didžiosiomis raidėmis parašytų santrumpų.

Būna problemų su iš svetur atėjusių simbolinių vardų linksniavimu. Kartais pridedama linksnį atitinkanti galūnė, atskirta apostrofu. Bet tai neestetiška ir ne visada tikslu. Tekstas bus sklandesnis prie simbolinio vardo pridėjus bendrinį objekto pavadinimą, pavyzdžiui,

Go back to Facebook → Grįžti į „Facebook“ tinklą;
Facebook friend → Draugas „Facebook“ tinkle.

Programų vardus ypač mėgstama vartoti jų pačių žinynuose ir kituose tekstiniuose dokumentuose. Tai galima paaiškinti autorių noru neišreikštiniu būdu reklamuoti savo gaminį, o gal ir teksto redagavimo stoka.

Technikos literatūros rašymo vadovuose sakoma, kad žodžių kartojimo reikia vengti. Jeigu programos vardas paminėtas antraštėje arba teksto pradžioje, tai toliau jau aišku apie ką kalbama ir nebereikia jo kartoti. Pavyzdžiui, dokumento (žinyno,

[1] Translate Toolkit & Pootle, http://translate.sourceforge.net/wiki/guide/translation/program_names

tinklalapio), frazes, kuriomis aprašomi „Skype“ pokalbių programos veiksmai, reikėtų performuluoti taip:

„Skype“ programa tikrina ryšį → Tikrinamas ryšys.
Atveriamas „Skype“ programos adresatų paieškos langas → Atveriamas adresatų paieškos langas.

Kas kita, jei reikia aprašomą programą atskirti nuo kitų programų, pavyzdžiui,

„Firefox“ naršyklė turi daugiau funkcijų negu „Internet Explorer“.
Išleista nauja „Firefox“ naršyklės laida.

Kai aišku apie kurią programą ar sistemą kalbama, bet reikia veiksnio, pakanka ją įvardyti bendriniu žodžiu: *programa, naršyklė, sistema* ir pan. Kai čia pat programą reikia paminėti pakartotinai, tai vietoj jos vardo arba bendrinio žodžio galima pavartoti įvardį.

Tekstuose, išskyrus antraštes bei oficialius paminėjimus, vardus galima trumpinti, pavyzdžiui, *Microsoft Office → MS Office, Internet Explorer → IE, Mozilla Firefox → Firefox → FF, Facebook → FB*.

Ar nebus nusižengimas prekės ženklo taisyklėms, jeigu trumpinami arba praleidžiami vardai, turintys prekės ženklo statusą? Prekės ženklai ir jų registravimas buvo sugalvoti prekėms žymėti ir suteikti išimtinę žymėjimo teisę ženklo savininkui. T. y. įstatymai riboja jų vartojimą, bet neįpareigoja jų vartoti ten, kur jais žymimas gaminys neturi prekės statuso.

Matavimo vienetai. Beveik visa originali programinė įranga parašyta anglų kalba. Todėl būna programų, kuriose operuojama tik colinės sistemos vienetais be galimybės juos pakeisti. Tai grubi internacionalizavimo klaida. Jeigu planuojama tokią programą lokalizuoti, tai apie šią klaidą reikia informuoti originalo autorius, kad ją ištaisytų prieš pradedant lokalizavimą.

Būna ir taip, kad programoje minimi konkretūs dydžiai, išreikšti coliais arba kitais nesisteminiais vienetais, pavyzdžiui: *...if left margin exceeds 2" then...*, tai tada reikia pakeisti matavimo vienetus ir perskaičiuoti dydžio reikšmę: *...jeigu kairioji paraštė didesnė negu 5,08 cm, tai....* Arba, jeigu tikslumas nesvarbu, – *5 cm*.

Lietuvių kalboje išvestiniai vienetai žymimi taip, kaip priimta Tarptautinėje vienetų sistemoje – pasviruoju brūkšniu bet kurios sistemos vienetams, pvz., km/val, Mb/s. Anglai vietoj pasvirojo brūkšnio dažnai rašo raidę *p*. Tokias santrumpas reikia pakeisti vartojamomis lietuvių kalboje (34 lentelė).

Pavyzdžiai. Tekstuose būna pavyzdžių, susijusių su kalba arba valstybe: asmenvardžių, vietovardžių, pašto adresų, piniginių vienetų ir pan. Jie turėtų organiškai įsilieti į lokalizacijos kalbą. Juos reikia ne perrašyti ar pažodžiui versti, o pakeisti būdingais lietuvių kalbai.

34 lentelė. Dažniau pasitaikančių vienetų (dimensijų santrumpų) vertimas

Vieneto pavadinimas		Santrumpa	
Anglų kalba	**Lietuvių kalba**	**Anglų kalba**	**Lietuvių kalba**
Bits per inch	Bitų colyje	bpi	b/col.
Bits per second	Bitų per sekundę	bps	b/s, b/sek.
Bytes per second	Baitų per sekundę	Bps	B/s, B/sek.
Dots per inch	Taškų colyje	dpi	tšk./col.
Frames per second	Kadrų per sekundę	fps	kadr./min.
Revolutions per second	Apsisukimų per minutę	rpm	aps./min.
Cycles per second, Hertz	Hercas	cps	Hz
Pixels per inch	Pikselių colyje	ppi	px/col.
Words per minute	Žodžių per minutę	wpm	žod./min.
Grams per square meter	Kvadratinio metro svoris gramais	gsm	g/m^2

Vietoj asmenvardžio *John Smith* lokalizacijoje derėtų pateikti kokį nors lietuvišką asmenvardį. Geriau tokį, kuriame yra savitųjų lietuvių kalbos raidžių, kad skaitytojui nekiltų abejonių dėl jų vartojimo asmenvardžiuose, pavyzdžiui, *Šarūnas Šarūnaitis, Jūratė Jūraitė* ir pan. Nereikėtų bijoti, kad pavyzdyje panaudotas asmenvardis sutaps su kurio nors asmens tikru asmenvardžiu.

Pašto adresą *john.smith@mail.com* derėtų pakeisti į *jjonaitis@abc.lt* ar panašų. Gaila, kad kol kas dar negalima visų lietuvių kalbos abėcėlės raidžių vartoti abonento varde (prieš ženklą @). Rimtų techninių priežasčių šiam ribojimui nėra. Tačiau esami protokolai nustato, kad čia gali būti tik anglų kalbos abėcėlės raidės. Šiuo metu (2010 m.) rengiami RFC dokumentai panaikinantys šį ribojimą. Tačiau tikrame asmenvardyje, rašomame prieš elektrononio pašto adresą, galima vartoti visas lietuvių kalbos (ir bet kurios pasaulio kalbos) abėcėlės raides, pavyzdžiui, *Jūratė Jūraitė <jrt@banga.lt>*.

Interneto adresų pavyzdžius galima keisti lietuviškais, pavyzdžiui, *http://www.yawl.com* galima būtų pakeisti į *http://ačiū.lt*. Čia vėlgi rekomenduotina panaudoti savitąsias lietuvių kalbos abėcėlės raides.

Pavyzdžiuose minimus vietovardžius, įstaigų pavadinimus ir kitokius su Lietuva nesusijusius pavyzdžius reikia pakeisti analogišką prasmę turinčiais lietuviškais.

6.2.5. Sujungtos eilutės

Programose pasitaiko vietų, kai ekrane matomos eilutės (frazės), sudaromos iš kelių, dažniausiai dviejų, išteklių eilučių, vartojamų ir kitur – savarankiškai arba kituose frazių junginiuose. Būdingas pavyzdys: užrašas (etiketė) prie mygtuko, atšaukiančio paskutinį įterpimo veiksmą (komandą *Insert*), angliškoje programos versijoje gali būti *Undo Insert*. Jeigu paskutinis buvo šalinimo veiksmas (komanda *Delete*), tai etiketės užrašas turėtų būti *Undo Delete*. Jeigu po atšaukimo veiksmo reikia grįžti į ankstesnę būseną, kuri buvo prieš atšaukimą, tai tada reikia pasinaudoti kitu mygtuku, prie kurio bus užrašas *Redo Insert* arba *Redo Delete*, žiūrint kas buvo atšaukta prieš tai. Bet lokalizuojamuose ištekliuose yra tik eilutės po vieną žodį: *Undo, Redo, Insert, Delete*. Atskiri žodžiai vartojami komandoms pažymėti, o suporuoti – komandoms atšaukti arba pakartoti (grįžti į tai, kas buvo atšaukta). Taip gaunama teksto ekonomija. Jeigu atšaukiamų ir grąžinamų komandų yra n, tai nenaudojant eilučių jungimo lokalizuojamuose ištekliuose reikėtų turėti 4×n žodžių, o naudojant – 2×n + 2 žodžius.

Visa tai buvo originalo programavimo dalykai. Tačiau jie turi įtakos ir vertimui. Tam, kad ekrane matytume taisyklingus lietuviškus užrašus *Atšaukti įterpimą, Atšaukti šalinimą, Grąžinti įterpimą, Grąžinti šalinimą*, operacijų pavadinimus (*Insert, Delete*, ...) reikia versti galininko linksniu ir rašyti iš mažųjų raidžių, nors iš angliškų eilučių to nematyti. Todėl šiuo atveju vertime dažniausiai daroma klaida (*Atšaukti Šalinti*, ...), kuri pastebima testuojant programą ir ištaisoma.

Anglų kalboje nėra galininko, o veiksmažodžio bendratis sutampa su veiksmažodinio daiktavardžio vardininku, tad originalo autoriai dažnai pamiršta, kad kitose kalbose gali būti kitaip. Dėl to tas pačias išteklių eilutes dažnai panaudoja ir pačioms komandoms įvardyti, kur kitose kalbose reikalinga veiksmažodžio bendratis. Tai internacionalizacijos klaida, nes visavertė lokalizacija nebeįmanoma. Jeigu išversime *Atšaukti* ir *Įterpti*, tai atskiriems mygtukams tiks, bet vietoj *Undo Insert* pasirodys užrašas *Atšaukti Įterpti*. O jei *Insert* išversime kaip *įterpimą*, tai ir ant atskiro įterpimo komandos mygtuko (ir galbūt ne vieno) pasirodys užrašas *įterpimą*. Nei šis, nei tas.

Apie klaidą reikia informuoti originalo autorių. Kadangi ši klaida dažniausiai išryškėja per vėlai, tai lokalizuojamoje programos laidoje autoriai gali nebespėti jos pataisyti. Tokiu atveju galima tik sušvelninti klaidos pasekmes tarp dviejų bendračių įterpus kokį nors skirtuką, pavyzdžiui: *Atšaukti | Įterpti*.

Tokia situacija būdinga ne tik lietuvių kalbai, bet ir kitoms sintetinėms kalboms. Problemų nekiltų, jeigu programos originalo kalba turėtų daugiau išraiškos priemonių, negu lokalizacijos kalba.

Lokalizuotojas gautų daugiau informacijos, jeigu originale žodžiai iš didžiųjų raidžių būtų rašomi tik sakinio pradžioje.

Kūrybiškai panaudojant gramatinių formų įvairovę (linksnius, kalbos dalių įvairovę) galima užglaistyti eilučių sandūras ir pasiekti, kad tekstas būtų vientisas, aiškus. Lietuvių kalba turi daugiau išraiškos priemonių, negu anglų. Dėl to yra didesnės galimybės padaryti, kad lietuviškas užrašas (pvz., *Atšaukti įterpimą*) lietuviui būtų suprantamesnis, negu anglui angliškas (*Undo Insert*).

Prie lokalizuojamų išteklių eilutės gali būti prijungta eilutė, suformuota programiniu būdu, dažniausiai skaičius. Pateiksime ekrane matomų eilučių pavyzdžių iš originalo programos ir kokios jos turėtų būti lietuviškoje lokalizacijoje:

Fig. 3 → 3 pav.;
Page 8 → 8 puslapis;
25 minutes → 25 minutės.

Pirmoji eilutė yra užrašas po iliustracija. Lietuviškame tekste numeris turėtų būti pradžioje. Bet čia galima išversti tik *Fig. → pav.*, bet nėra galimybės šį žodį sukeisti vietomis su skaičiumi. Ekrane bus matomas užrašas *pav. 3*.

Tas pats pasakytina ir apie antrąjį pavyzdį.

Tai internacionalizacijos klaidos. Originalo autoriai gali jas pataisyti panaudodami parametrizuotą eilutę (6.2.6 skyr.).

Trečiojo pavyzdžio vertimui reikia trijų žodžio formų: *minutė*, *minutės* ir *minučių*. Programa turi parinkti prie skaičiaus deramą formą ir ją perduoti į ekraną. Todėl šiuo atveju originalas turi turėti priemones daiktavardžių daugiskaitos formoms parinkti (6.2.7 skyr.).

6.2.6. Parametrizuotos eilutės

Eilutėje be tikro teksto gali būti parametrų. Tai kitų eilučių vardai, kurie programos vykdymo metu pakeičiami jais įvardytomis eilutėmis, t. y. tikrais jų tekstais. Parametras iš tikro teksto išskiriamas (atpažįstamas) įvairiai: prieš jį rašomas koks nors sutartas ženklas (pvz., %, #), parametras rašomas tarp dviejų ženklų (pvz., %...%, {...}) ir pan.

6.2.5 skyrelyje pateiktą užrašo po iliustracija formavimo eilutę *Fig*. papildysime parametru:

Fig. %d

Dabar lokalizacijoje jau galima eilutės komponentus sukeisti vietomis:

%d pav.

Parametras gali būti bet kurioje eilutės vietoje, pavyzdžiui,

Remain %d min.

Liko %d min.

Kiek minučių liko iki kurio nors proceso pabaigos galima apskaičiuoti tik programos vykdymo metu. Besikeičianti parametro reikšmė nuolat perskaičiuojama ir įterpiama į ekrane matomą pranešimą vietoj *%d*.

Parametras gali būti sudarytas iš kelių ženklų (raidžių, skaitmenų), vienoje eilutėje gali būti keli parametrai, pavyzdžiui,

%title is played by %artist

Kūrinį „%title“ atlieka %artist

Antruoju atveju vietoj parametrų įrašomas kūrinio pavadinimas (lietuviškame vertime tarp kabučių) ir jo atlikėjas. Ilgesni parametrų vardai vartojami tada, kai tas pats parametras panaudojamas daugelyje eilučių. Parametrų vardai neverčiami, nes jie ekrane nematomi.

Toje pačioje eilutėje esantys keli parametrai gali būti pavadinti tuo pačiu vardu. Tada jie keičiami konkrečiomis eilutėmis iš eilės: pirmoji gauta reikšmė įrašoma vietoj pirmojo parametro, antroji – vietoj antrojo ir t. t. Tai paprasta, bet tada lokalizuotoje eilutėje negalima parametrų keisti vietomis. O keisti prireikia, nes žodžių tvarka įvairių kalbų sakiniuose gali skirtis.

Jeigu parametrai būtų pažymėti tais pačiais vardais, tai to padaryti nebūtų galima. Tai originalo trūkumas, laikytinas internacionalizacijos klaida. Net ir tuo atveju, kai lietuviškame vertime jų eilės keisti nereikia – galbūt jų išdėstymo eilė netiks kuriai nors kitai kalbai.

Kartais po parametro vardo būna prirašyta raidė *s* arba *'s*, pavyzdžiui, *%1s*, *%2's*. Tai reiškia, kad angliškoje programoje ši galūnė bus pridedama prie parametro, pavyzdžiui, jeigu parametrą keičiantis žodis yra *file*, tai jis virs į *files* arba *file's*, kas reikš daugiskaitą arba vienaskaitos kilmininką. Kitoms kalboms tai netinka ir daugiskaitos arba kilmininko nepavyks padaryti. Toks atvejis yra internacionalizacijos klaida ir apie ją reikia pranešti originalo autoriui, kad ištaisytų.

Ženklų, vartojamų parametrams išskirti, gali prireikti ir pagal tiesioginę jų paskirtį, pavyzdžiui, ženklo % procentams žymėti. Tokiu atveju jie atskiriami pagal kontekstą (pvz., jei po % eina tarpas, tai jis žymi procentus, jei raidė arba skaitmuo – parametro pradžią), ženklą pakartojant du kartus (pvz., vietoj 99 % rašoma 99 %%) arba kokiu nors kitokiu būdu.

Kai parametras simbolinis vardas, tai jo faktinė reikšmė ekrane turi būti rodoma tarp kabučių. Kartais kabutės būna ir originale. Bet kokiu atveju jas reikia pakeisti

lietuvių kalboje vartojamomis kabutėmis, o jei jų originale nėra – į lokalizuojamą programą jas įdėti, pavyzdžiui,

Installing \""%A\"" to \""%B\"" → Diegiama „%A“programa į „%B“ aplanką.

Installing "%A\" to "%B" → Diegiama „%A“ programa į „%B“ aplanką.

Installing %A to %B → Diegiama „%A“ programa į „%B“ aplanką.

Kartais reikia eilutę pertvarkyti:

Page %d of %d → %d iš (%d)

Panaudojant parametrų mechanizmą gali būti užrašytas datos formatas:

%d-%m-%Y %H:%M → %Y-%m-%d %H:%M

čia *d* – diena, *m* – mėnuo, *Y* – metai, *H* – valandos, *M* – minutės.

6.2.7. Gramatinių formų derinimas

Kai po skaičiaus eina su tuo skaičiumi susijęs žodis, tai jo linksnis priklauso nuo skaičiaus (1 daiktas, 2 daiktai, 10 daiktų). Kai skaičius nežinomas, popieriniuose dokumentuose išvardijami visų galimų jo linksnių galūnės (... daiktas(-ai, -ų)). Skaičius sužinomas tik programos vykdymo metu ir gali keistis. Pagal jį programa turi nustatyti, kuri žodžio forma tuo metu reikalinga, ir ją pateikti į ekraną arba išspausdinti į dokumentą. Tuo nepasinaudoti ir vardyti visus galimus atvejus kompiuteriniuose tekstuose taip, kaip popieriniuose dokumentuose, reikštų nesugebėjimą pasinaudoti kompiuterio teikiamomis galimybėmis.

Lietuvių kalba turi tris formas: vieną vienaskaitos (1 daiktas) ir dvi daugiskaitos (2 daiktai, 10 daiktų). Išeitų, kad lietuviams nuo 2 iki 9 yra *keli daiktai*, o virš 10 – *daug daiktų*.

Vienaskaitos forma vartojama ir su daugeliu daiktų, jeigu jų skaičius baigiasi vienetu (21 daiktas, 31 daiktas, ... 101 daiktas...), išskyrus 11, nes derinamas su prieš tai einančiu žodžiu *vienas*.

Anglų kalba turi dvi formas: vieną vienaskaitos ir vieną daugiskaitos. Vienaskaitos forma vartoja su vieninteliu skaičiumi – vienetu. Jeigu didesnis skaičius baigiasi vienetu, tai vis tiek vartojama daugiskaita.

Tokias pačias formas turi ir vokiečių kalba. Tik vokiečiams iš dalies pavyksta suderinti gramatines formas: kai dešimtys baigiasi vienetu, tai, skaitydami skaičių, vienetą atkelia prieš dešimtis, pavyzdžiui, *21 → ein und zwanzig*.

Programos lokalizacijoje turi būti nurodytos visos po skaičiaus rašomo daiktavardžio formos, o programos originale – mechanizmas reikiamai formai nustatyti iš dinamiškai besikeičiančio skaičiaus. Kadangi beveik visos originalios programos būna angliškos, o anglų kalba turi tik dvi labai paprastai nustatomas formas, tai neretai pamirštamos kalbos, turinčios daugiau formų. Tai internacionalizacijos klaida.

Pailiustruosime, kaip su daiktavardžių formomis tvarkomasi PO formato lokalizuojamuose ištekliuose.

Kiekvieno failo, kuriame bus tokių formų, pradžioje, įvadiniame lokalizavimo informacijai skirtame įraše, būna tokio pavidalo eilutė:

"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"

Vietoj žodžio INTEGER įrašomas formų skaičius, o vietoj EXPESSION Javos programavimo kalba užrašytas reiškinys formos numeriui iš skaičiaus *n* nustatyti. Lietuviškoje lokalizacijoje tai būtų:

"Plural-Forms: nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && (n%100<10 or n%100>=20) ? 1 : 2)"

Kintamuoju *nplurals* įvardytas visų formų skaičius, nors angliškai *plural* reiškia daugiskaitą.

Tam, kad užrašas būtų vaizdesnis, užrašysime jį matematiškai:

$n \backslash 10 = 1$ ir $n \backslash 100 \neq 11 \rightarrow 0$

$n \backslash 10 \geqslant 2$ ir ($n \backslash 100 < 10$ arba $n \backslash 100 \geqslant 20$) $\rightarrow 1$

$\rightarrow 2$

čia ženklu \ žymėjome operaciją, kurios rezultatas yra dalybos liekana.

Toliau visi įrašai, kuriuose yra prie skaičių derinamų daiktavardžių, programos originale pateikiami tokiu pavidalu:

```
msgid "%d minute"
msgid_plural "%d minutes"
msgstr[0] ""
msgstr[1] "
```

Procento ženklu pažymėtas parametras – vieta eilutėje, kur bus įstatomas minučių skaičius.

Lietuviškoje lokalizacijoje šis įrašas papildomas dar viena eilute trečiai formai įrašyti ir atrodytų taip:

```
msgid "%d minute"
msgid_plural "%d minutes"
msgstr[0] "%d minutė"
msgstr[1] "%d minutės"
msgstr[2] "%d minučių"
```

Pagal tokią pačią schemą gali būti užrašyti ir kiti linksniai, pavyzdžiui:

msgid "%d minute ago"
msgid_plural "%d minutes ago"
msgstr[0] "prieš %d minutę"
msgstr[1] "prieš %d minutes"
msgstr[2] "prieš %d minučių"

Gali būti ir kitokių variantų. Pavyzdžiui, KDE operacinės sistemos realizacijoje yra keturios formos. Papildoma forma skirta tikram vienetui (ne 21, 31, ...), kad derėtų su idėja vienetą rašyti žodžiu, pvz., *viena minutė*.

Kitokių formatų išteklių failuose daugiskaitos formos nurodomos kitaip. Tačiau rezultatas, matomas ekrane, nuo to nepriklauso.

Sudėtingiau, kai linksniuojamas žodis lokalizavimo metu nežinomas. Būdingas pavyzdys – kliento vardas kurioje nors paslaugų sistemoje. Asmuo, registruodamasis į ją, pateikia savo vardą vardininko linksniu, pvz., *Jonas*. Kai programa kreipiasi į jį, pavyzdžiui, *Sveiki, Jonai*, reikalingas šauksmininkas, kurį turi suformuoti programa – negi prašysi kliento, kad registruodamasis pateiktų savo vardą dar ir šauksmininko linksniu. Šauksmininko forma nevienoda (*Jonas → Jonai, Marius → Mariau, Rytis → Ryti, Jūratė → Jūrate, Ona → Ona*). Priklauso nuo vardo linksniuotės.

Socialinių tinklų, pokalbių ir kitose programose, kuriose vyksta įvairiaplanis bendravimas tarp žmonių, reikalingi visi linksniai. Ir ne tik lietuvių kalboje. Pailiustruosime tai pavyzdžiu iš „Pidgin“ programos (35 lentelė).

Su asmenimis vartojami visi linksniai, išskyrus vietininką. Tačiau pokalbių kambariui įvardyti prireikė ir vietininko. Taigi faktiškai reikalingi visi linksniai.

Įvairių linksnių formos reikalingos visose sintetinėse kalbose. Todėl tokia galimybė turi būti programose, kurias tikimasi lokalizuoti bent vienai sintetinei kalbai. Kitaip – internacionalizavimo klaida.

Pokalbių programose, dar dažniau socialiniuose tinkluose, nurodoma asmens būsena išreiškiama būtojo kartinio laiko veikiamuoju dalyviu: *prisijungęs(-usi), pasitraukęs(-usi), užimtas(-a), išvykęs(-usi), vedęs(-usi)* ir pan. Čia reikia dinamiškai parinkti asmenį atitinkančią dalyvio giminę:

Jonas prisijungęs,
Jonienė prisijungusi.

Dalyviai pagal giminę kaitomi sintetinėse kalbose. Jose asmens lytį beveik visada galima nustatyti iš asmenvardžio. Tai tinka vienos kalbos arba kelių, tokią pat savybę turinčių, kalbų terpėje.

35 lentelė. Linksnių vartojimas „Pidgin“ programoje anglų, lietuvių, rusų ir lenkų kalbomis

Linksnis	Anglų	Lietuvių	Rusų	Lenkų
Vardininkas	John	Jonas, Marius, Rytis, Jūratė, Ona	Иван, Дмитрий, Игорь, Фадей, Татьяна, Любовь, Прасковья, Евдокия, Майя	Jan, Antoni, Regina, Mariusz
Kilmininkas	Change password for John. The email address for John is ...	Keisti Jono (Mariaus, Ryčio, Jūratės, Onos) slaptažodį. Jono (Mariaus, Ryčio, Jūratės, Onos) el. pašto adresas yra ...	Изменить пароль Ивана (Дмитрия, Игоря, Фадея, Татьяны, Любови, Прасковьи, Евдокии, Майи). Адрес электронной почты Ивана (Дмитрия, Игоря, Фадея, Татьяны, Любови, Прасковьи, Евдокии, Майи): ...*	Zmieniam parol Jana, Antoniego, Reginy, Mariusza
Naudininkas	You are sending messages too fast to John. Sending To: John	Jonui (Mariui, Ryčiui, Jūratei, Onai) siunčiate per sparčiai. Siunčiama Jonui (Mariui, Ryčiui, Jūratei, Onai)	Вы отправляете сообщения Ивану (Дмитрию, Игорю, Фадею, Татьяне, Любови, Прасковье, Евдокии, Майе) слишком часто. Отправка Ивану (Дмитрию, Игорю, Фадею, Татьяне, Любови, Прасковье, Евдокии, Майе)	Przesylam dane Janu, Antoniemu, Reginie, Mariuszowi
Galininkas	Peter wants to add John to his buddy list	Petras nori įtraukti Joną (Marių, Rytį, Jūratę, Oną) į savo bičiulių sąrašą	Пётр хочет добавить Ивана (Дмитрия, Игоря, Фадея, Татьяну, Любовь, Прасковью, Евдокию, Майю) в свой список контактов	Piotr chce dołączyc Jana, Antoniego, Reginę, Mariusza do listy kontaktów
Įnagininkas	Conversation with John	Pokalbis su Jonu (Mariumi, Ryčiu, Jūrate, Ona)	Беседа с Иваном (Дмитрием, Игорем, Фадеем, Татьяной, Любовью, Прасковьей, Евдокией, Майей)	Rozmawiam z Janem, Antonim, Reginą, Mariuszem
Vietininkas	Chat in Meeting is not available	Pokalbis „Pasimatyme“ negalimas {„Pasimatymas“ – kambario vardas}	Чат в «Свидании» не доступен	---
Šauksmininkas	Hello John. Dear John	Sveiki, Jonai (Mariau, Ryti, Jūrate, Ona). Gerb. Jonai (Mariau, Ryti, Jūrate, Ona)	Добро пожаловать, Иван (Дмитрий, Игорь, Фадей, Татьяна, Любовь, Прасковья, Евдокия, Майя)!	Witam cię Janie, Antoni, Regino, Mariuszu

* Vietoj kilmininko rusų kalboje gali būti naudojamas būdvardis (притяжательное прилагательное), nurodantis daikto priklausomybę asmeniui ir sudaromas atsakant į klausimą kieno? (чей?) Pavyzdžiui, Иванов (Дмитриев, Игорев, Фадеев, Татьянин, Любовин, Прасковьин, Евдокиин, Майин) пароль; Иванова (Дмитриева, Игорева, Фадеева, Татьянина, Любовина, Прасковьина, Евдокиина, Майина) учетная запись. Tačiau šiuolaikinėje rusų kalboje dažniau vartojamas kilmininko linksnis.

Universalus problemos sprendimas, tinkantis ir daugiakalbei terpei, būtų giminę nustatyti iš asmens registracijos duomenų – registruojantis į tokius tinklus paprastai prašoma nurodyti lytį. Belieka pareikalauti, kad toks anketos laukas būtų privalomas, ir problema bus išspręsta globaliai.

Rusų, lenkų ir kai kurių kitų kalbų būtojo kartinio laiko trečiojo asmens veiksmažodžiai turi giminę ir skaičių, pavyzdžiui, žodis *rašė (писал),* rašomas prieš citatą, rusų kalboje turi tris formas:

Алексей писал
Татьяна писала
Татьяна и Алексей писали

Elektroniniame pašte šią problemą būtų galima išspręsti tik vienos arba kelių panašių kalbų terpėje. Globaliniam sprendimui nėra informacijos apie asmens lytį laiško antraštėje. Bet pokalbių programose ir socialiniuose tinkluose tokia informacija yra registracijos duomenyse ir problema nesunkai išsprendžiama.

Kartais tenka parinkti tinkamą įvardžio giminę ir skaičių priklausomai nuo to, kokios giminės asmenį(-is) jis pakeičia, pavyzdžiui,

Jis grįš po valandos,
Ji grįš po valandos,
Jie grįš po valandos.

Socialiniuose tinkluose žinučių ir įvairių pranešimų tekstai dažnai formuojami iš atskirų frazių. Todėl čia gramatinių formų derinimas ypač svarbus ir sudėtingas. Tai atskira plati tema, ties kuria čia nesustosime.

6.2.8. Terminija

Beveik visa programinė įranga verčiama iš anglų kalbos. Tačiau įvairūs autoriai vartoja skirtingą terminiją ir leksiką. Pateiksime būdingesnių atvejų.

Anglų kalboje dažni atvejai, kai tuo pačiu žodžiu įvardijamos skirtingos sąvokos, pavyzdžiui,

check → *pažymėti* (varnele); *tikrinti* (pvz., rašybą);
From ... To → *Kas ... Kam* (el. laiške); *nuo ... iki*;
sort → *rikiuoti; rūšiuoti;*
tab → *kortelė; qselė, tabuliavimo ženklas;*
title → *antraštė* (dokumente); *titulas* (asmens);
user →*naudotojas, vartotojas, skaitytojas, lankytojas;*

Dažnai daromos klaidos verčiant angliškus žodžius, kurie panašūs į lietuvių kalboje vartojamus tarptautinius žodžius, turinčius kitokią prasmę, pavyzdžiui,

theme → *apvalkalas,* programos sąsajoje;
subject → *tema,* el. laiško antraštėje;
paragraph →*pastraipa*;
paragraph → *paragrafo ženklas*;
Painiojami panašios rašybos žodžiai, pavyzdžiui,
Finish → baigti (Finish ≠ Finnish → suomių);
Kannada → kanadų (Kannada ≠ Canada → kanadiečių);
Italic → kursyvas (Italic ≠ Italian → italų);
Dominica → Dominikos Respublika (Dominica ≠ Dominican → Dominikonų).

Vienodos daiktavardžių ir veiksmažodžių formos. Daugelis tą pačią formą turinčių (vienodai rašomų) angliškų žodžių gali būti ir daiktavardžiai, ir veiksmažodžiai:

backup → *archyvuoti; atsarginė kopija;*
bookmark → *adresas* (tinklalapių adresyno įrašas), *įrašyti adresą* (į adresyną);
display → *monitorius* (kompiuterio įtaisas), *rodyti* (pvz., tam tikra koduote ir šriftu);
file → *failas; įdėti* (ką nors į aplanką);
list → *sąrašas; išvardyti.*

Sudėtinių terminų trumpinimas. Kai iš konteksto aišku, apie ką kalbama, sudėtinio termino pažymimuosius žodžius galima praleisti, pavyzdžiui,

standusis diskas → diskas;
kompaktinis diskas → diskas;
daugkartinio rašymo kompaktinis diskas → daugkartinio rašymo diskas;
vienkartinio rašymo kompaktinis diskas → vienkartinio rašymo diskas;
disko takelis → takelis;
disko sektorius → sektorius;
absoliučiosios langelio koordinatės → langelio koordinatės;
absoliutusis universalusis adresas → universalusis adresas;
centrinis procesorius → procesorius;
tekstų rengyklė → rengyklė;
universalusis adresas → adresas;
elektroninis laiškas → laiškas;
nereikšminių nulių atmetimas → nulių atmetimas.

6.2.9. Pasinaudojimas kitų kalbų vertimais

Jeigu programa išversta į kitas kalbas, kurias supranta lokalizuotojas, naudinga pasitelkti jos vertimus. Kartais tik kita kalba padeda suprasti, kokią prasmę suteikė programos originalo autorius kuriam nors angliškam žodžiui.

Grožinę literatūrą rekomenduojama versti iš originalo, nes vertimas negali prilygti originalui. Tačiau yra adaptuotų programinės įrangos vertimų, pranokstančių originalą. Todėl šią situaciją verta išnaudoti.

Bet kuriuo atveju į kitų kalbų vertimus reikia žiūrėti kritiškai, nes būna ir prastų vertimų. Iš jų reikia atsirinkti tik tai, kas gera, neperkelti klaidų iš jų į savo vertimus.

6.3. Lokalizuojamos programos testavimas

Kompleksinis lokalizuotos programos testavimas atliekamas užbaigus visus programos vertimo ir adaptavimo darbus. Kompleksinį testavimą tikslinga suskirstyti į dvi dalis: 1) vidinį testavimą – atlieka ta pati grupė, kuri lokalizavo programą arba šiam darbui įpareigoti recenzentai; 2) išorinį – atlieka kiti asmenys, specialiai testuojantys programą arba ją naudojantys kasdieniniame darbe ir informuojantys lokalizuotojus apie pastebėtas klaidas.

Su programinės įrangos lokalizavimu ir lokalizuotos programos testavimu glaudžiai susijusios sąvokos *internacionalizavimo klaida* ir *lokalizavimo klaida*.

Internacionalizavimo klaida laikysime programinės įrangos projektavimo klaidą, kai tam tikra jos savybė arba parametras nenumatytas įvairioms lokalėms pritaikyti. Internacionalizavimo klaidų turinti programinė įranga nėra neutrali kultūriniu ir kalbiniu požiūriais, todėl ji negali būti laikoma visiškai internacionalizuota. Ją sudėtinga ar net neįmanoma (priklausomai nuo klaidų skaičiaus ir sudėtingumo) lokalizuoti. Internacionalizavimo klaidų pavyzdžiai: grafinės sąsajos tekstai ir pranešimai neatskirti nuo pirminio programos teksto; trupmenos skirtukas, laiko formatas, pirmoji savaitės diena ir kiti parametrai įkoduoti ir jų negalima pakeisti nei diegimo metu, nei įdiegtoje programoje.

Dažniau pasitaikančių internacionalizavimo klaidų sąrašas pateiktas 5 priede. Tai pagrindiniai klausimai, į kuriuos reikėtų atsakyti testuojant programą.

Lokalizavimo klaida – tai programinės įrangos kurios nors savybės neatitikimas lokalei, kai ta savybė yra internacionalizuota. Pavyzdžiui, lokalizavimo klaida gali būti laikoma netinkamai parinkta lokalizuotos programos numatytoji koduotė, kai ta koduotė yra įtraukta į programą, netinkamas datos arba laiko formatas, kai programoje yra galimybė jį pasirinkti, netinkami dokumentų šablonai ir kt. Netaisyklingas programinės įrangos tekstų ir pranešimų, matomų ekrane, vertimas ir komponavimas, netinkamų terminų parinkimas taip pat priskiriami prie lokalizavimo klaidų.

Dalį internacionalizavimo klaidų galima aptikti parengiamojo etapo metu, dalis jų randama lokalizuojant programą. Internacionalizavimo klaidas paprastai taiso programos autoriai. Kartais – patys lokalizuotojai (pvz., kai lokalizuojama atviroji programinė įranga).

Yra sukurtos programos formaliam vertimų patikrinimui: ar tame pačiame lange matomos pabrauktos meniu punktų raidės visos yra skirtingos, ar dialogo langų užrašai telpa į jiems skirtas vietas ir pan. (7.6 skyr.). Todėl šį darbą galima automatizuoti. Tačiau kol kas žmogui tenka atlikti svarbiausią testavimo darbų dalį: nustatyti, ar tinkamai išversti tekstai, ar lengvai ir vienareikšmiškai jie suprantami.

Testavimo metu išryškėja vertimo netikslumai, kurių nebuvo galima numatyti verčiant tekstus, nes daugelis angliškų žodžių yra nevienareikšmiai. Dažnai tas pats angliškas žodis (frazė) išteklių failuose yra kartojamas kelis kartus. Tai reiškia, kad skirtingi to paties žodžio egzemplioriai patenka į skirtingus kontekstus, todėl gali turėti skirtingus lietuviškus atitikmenis. Tokiu atveju reikia įsitikinti, ar išversta arba pataisyta frazė pateko ten, kur reikia. Sudėtingiau, kai tas pats žodžio arba frazės egzempliorius įvairiuose kontekstuose vartojamas skirtingomis prasmėmis arba turi turėti skirtingas gramatines formas. Tada reikia (ir kartais būna nelengva) parinkti frazę tokią, kad ji tiktų visiems kontekstams.

Išvertus visas išteklių failų frazes, visi ekrane matomi užrašai turėtų būti išversti. Deja, taip būna ne visada. Pasitaiko frazių, įdėtų tiesiogiai į programų (Java, JavaScript, C++, Perl ir kt.) tekstus. Reikia jas surasti ir išversti.

Originalo ir vertimo frazių ilgis dažnai būna skirtingas. Dėl to kartais tenka koreguoti ir dialogo langų dydį bei struktūrą.

Reikia suderinti linksnius, pakeisti gramatines formas (veiksmažodį daiktavardžiu, veiksmažodžio rūšį ir pan.). Visa tai galima atlikti tiktai realiai dirbant su lokalizuojama programa.

Pagal testavimo rezultatus programa koreguojama. Koreguojant atsiranda naujų klaidų bei nesuderinamumų. Todėl testavimo ir koregavimo ciklą tenka kartoti, kol klaidų nebelieka arba lieka mažai.

O'Sullivanas (O'Sullivan, 2001) išskiria tokias lokalizacijos testavimo sritis:

1. ***Grafinės sąsajos elementų darna.*** Lokalizuotoje programoje turi būti išlaikytas atitikimas originalios programos sąsajai (originale esantys sąsajos elementai turi būti ir lokalizacijoje).
2. ***Funkcionalumas***. Lokalizavimo metu neturi būti pažeistos programos funkcijos.
3. ***Vaizdo estetika***. Grafinių elementų išdėstymas, grupavimas, lygiavimas bei teksto išdėstymas grafiniuose elementuose. Tikrinama, ar nėra valdiklių ar teksto, kurie netelpa jiems skirtoje ekrano vietoje.
4. ***Sąsajos su kitomis programomis***. Jei originali programa naudojasi kitų programų paslaugomis, tai tikrinamas šių sąsajų išlaikymas adaptavus programą.
5. ***Scenarijų testavimas***. Jei į programą yra įtraukti scenarijai, parašyti, pvz., „Basic", „JavaScript" kalbomis, tai tikrinamas šių scenarijų veikimas lokalizacijoje.
6. ***Pranešimai apie klaidas***. Ankstesni tyrimai (O'Sullivan, 1999) parodė, kad apie 35 proc. programos lokalizuojamųjų išteklių eilučių yra pranešimai apie klaidas, todėl svarbu sukelti klaidų situacijas ir išbandyti, ar pranešimai atitinka klaidas.
7. ***Diegimo programa.*** Jei programinė įranga turi diegimo programą, tai ji testuojama taip pat, kaip ir pati programa.
8. ***Daugiaplatformis veikimas.*** Jei programa skirta kelioms platformoms (pvz., operacinėms sistemoms), tai testuojama kiekvienos platformos lokalizuotoje versijoje (jei yra lokalizuota).
9. ***Aparatinės įrangos testavimas.*** Testuojama su įvairia aparatine įranga, kuria naudojamasi lokalizuotoje programoje (įvairūs spausdintuvų, modemų, serverių, protokolų ir t. t. modeliai).
10. ***Kitkas***. Testai, priklausomi nuo programinės įrangos specifikos.

O'Sullivanas visuose išskirtuose testavimo darbuose pabrėžia lokalizacijos identiškumą originalui funkciniu, estetiniu ir kitais požiūriais. Tačiau realizuojant trečiąjį lokalizavimo laipsnį (1.2 skyr.) gali neišvengiamai pasikeisti grafinė sąsaja ir kai kurios funkcijos, siekiant kuo geriau pritaikyti konkrečiai programinei terpei. Todėl visa tai reikia turėti omenyje kompleksinio testavimo metu.

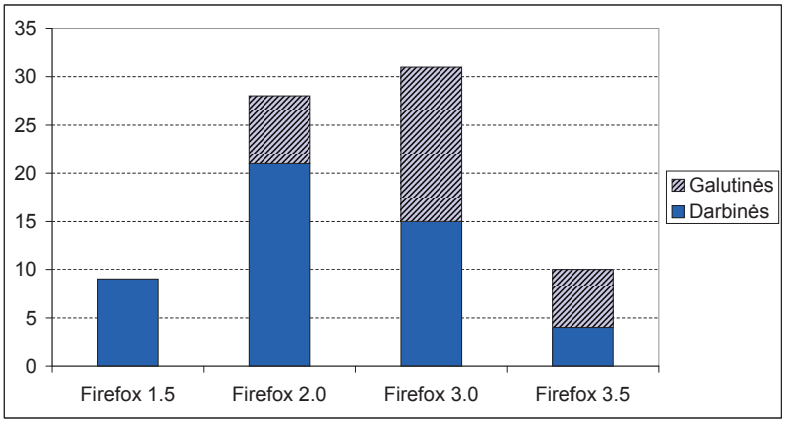
6.4. Lokalizavimo tęstinumas

Programos tobulinamos, atsiranda naujų versijų. Tam, kad lokalizuota programos atmaina būtų gyvybinga, reikia operatyviai atnaujinti jos lokalizuotas versijas, taisyti pastebėtus netikslumus, atsakyti į naudotojų klausimus.

Komercinės programos atnaujinamos maždaug kasmet. Atvirosios – dažniau. Kartu reikia atnaujinti ir lokalizacijas. Idealiu atveju originali ir lokalizuota versija turėtų pasirodyti vienu metu. Tai galima įgyvendinti esant nuolatiniam ryšiui su programos originalo autoriumi. Programos bandomąsias versijas ir lokalizavimo išteklių failus iš autoriaus galima gauti iš anksto ir juos lokalizuoti. Paskutiniai bandomųjų versijų koregavimai paprastai neturi įtakos programos lokalizavimui, išskyrus jos perkompiliavimą.

Internetinės programinės įrangos naujos versijos išleidžiamos dažnai, kartais atnaujinama internetu, automatiškai. Dėl to internetinės įrangos lokalizacijų išleidimas vienu metu su originalu yra beveik privalomas reikalavimas.

Pailiustruosime interneto naršyklės „Mozilla Firefox“ pavyzdžiu (remiantis duomenimis, skelbiamais „Mozilla.org“ oficialioje svetainėje). Per ketverius metus (nuo 2005 m. rudens – iki 2009 m. rudens) buvo išleistos 78 versijos. Iš jų 4 – pagrindinės versijos (kuriose yra daugiausia funkcinių pakeitimų ir jos platinamos galutiniam naudotojui), 45 pagrindinių versijų modifikacijų (kurios taip pat laikomos galutinėmis versijomis ir platinamos naudotojams, tačiau turi mažesnį skaičių funkcinių pakeitimų) ir 29 darbinių (alfa, beta, RC1, RC2 ir RC3) versijų, kurios skirtos testavimui, o ne galutiniam naudotojui (49 pav.).



49 pav. 2005–2009 m. išleistų naršyklės „Mozilla Firefox“ versijų pasiskirstymas

Lokalizavimo požiūriu dauguma darbų atliekama rengiant pagrindines versijas ir darbines versijas, kurių nuo 2005 iki 2009 metų buvo 29.

Kyla klausimas, kaip užtikrinti lokalizavimo tęstinumą, laiku, kartu su originaliomis, išleidžiant ir lokalizuotas versijas.

Kai kurios programos, pvz., internetinės bibliotekos, turi vidinį vertimo komponentą, kuriame eilutės yra padalintos į keletą grupių pagal naudojimo dažnumą. Pirma, lokalizuojamos dažniausiai naudojamos eilutės. Tačiau tai, be abejo, mažiau skatina lokalizuotojus pateikti visų išteklių lokalizaciją (Nichols ir kt., 2005).

Literatūroje galima rasti nemažai šaltinių, skirtų lokalizavimo ir vertimo automatizavimo priemonėms (Esselink, 2000; Bargary, 2006; Musale, 2004; Guzman, 2004; Wassmer, 2004 ir kt.). Šios priemonės padeda kaupti vertimus ir terminiją specialiose duomenų bazėse, importuoti ankstesnių versijų lokalizuotus išteklius į naujas versijas (žr. 7 skyrių). Tačiau yra atlikta tyrimų, kuriais patvirtinama, kad automatizuojant atnaujinimo procesą gali nukentėti darbo kokybė (Bowker, 2005). Be to, galimybė efektyviai pritaikyti tokias priemones priklauso nuo lokalizuojamųjų išteklių pateikimo metodo programoje (5.2 skyr.) ir nuo programos naudojamo lokalės modelio (4.3 skyr.).

7. LOKALIZAVIMO PRIEMONĖS

Lokalizuotojui visų pirma reikalinga bendroji informacija, kurią galima rasti vadovėliuose, žinynuose, žodynuose.

Lokalizavimo procesui palengvinti ir tam tikriems, daugiausia rutininiams, lokalizavimo veiksmams iš dalies automatizuoti naudojamos įvairios programinės priemonės.

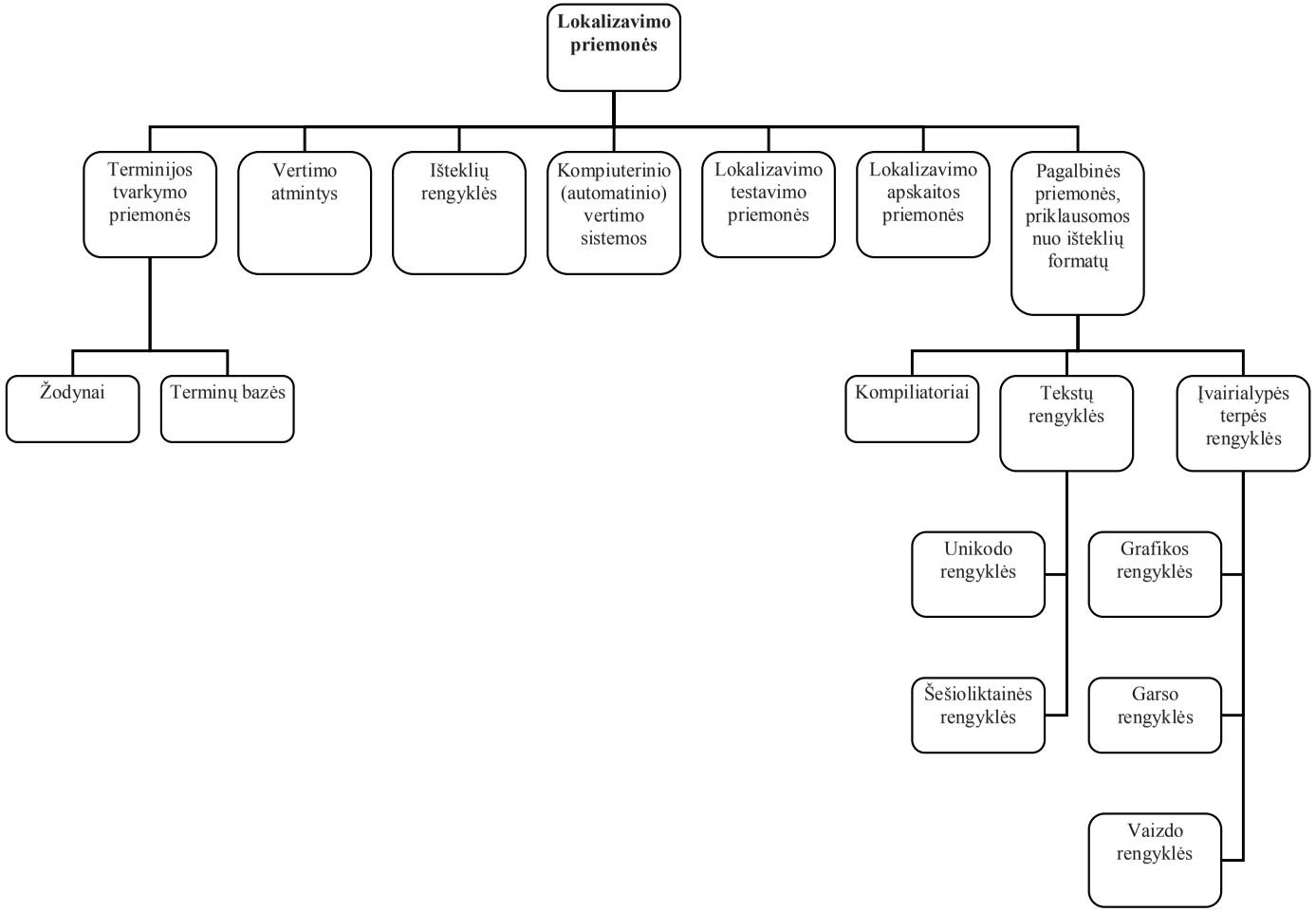
Lokalizavimui skirta programinė įranga ištraukia iš programos arba svetainės lokalizuojamuosius išteklius ir vaizdžiai juos pateikia, leidžia parinkti lokalės koduotę ir kitus lokalės elementus, suskirsto lokalizuojamą tekstą į segmentus, sudaro sąlygas sukurti, importuoti, eksportuoti, naudoti vertimo atmintį, terminų bazes, rašyti komentarus apie vertimus ir žymėti vertimus įvairiais sutartiniais ženklais (pvz., tikslintina eilutė), testuoti lokalizuotą programą ir kt.

7.1. Lokalizavimo priemonių struktūra

Lokalizavimo priemonės pradėjo rastis neseniai, pačioje XX a. pabaigoje, ir nuolat keičiasi. Galima klasifikacija pateikta 50 pav.

Labiausiai reikalingos priemonės apjungiamos į lokalizavimo priemonių paketus, vadinamus kompiuterizuoto vertimo (lokalizavimo) paketais. Dauguma tokių paketų apima įvairių formatų išteklių tvarkytuves, vertimo atmintis, terminų bazes, lokalizavimo apskaitos priemones, kartais – kompiuterinio vertimo, testavimo, taškinės grafikos modifikavimo priemones.

Nekeliame tikslo čia aprašyti konkrečias lokalizavimo priemones ar paketus, kadangi jie nuolat keičiasi ir tobulėja, apžvelgsime pagrindines priemonių rūšis, svarbiausias jų savybes ir taikymo lokalizavimo procese aspektus.



50 pav. Lokalizavimo priemonių schema

7.2. Žodynai

Pirminis žodžių ir terminų šaltinis yra žodynai: dvikalbiai, daugiakalbiai, aiškinamieji, enciklopediniai. Pirmasis stambesnis Lietuvoje išleistas žodynas buvo trikalbis rusų–lietuvių–anglų kalbų skaičiavimo technikos terminų žodynas (Kairys, V. ir kt., 1971). Po ilgos pertraukos iš dalies tie patys autoriai parengė didelį (apie 18 tūkst. žodžių) keturkalbį lietuvių–anglų–rusų–vokiečių kalbų informatikos terminų žodyną (Valatkaitė R., Kudirka Z., 1997). Per tą laiką buvo išleista ir keletas mažesnių anglų–lietuvių kalbų žodynėlių, tačiau šie du laikytini pagrindiniais, kuriuos kiti tik papildydavo.

Plintant asmeniniams kompiuteriams, ypač pradėjus lokalizuoti programinę įrangą, atsirado nauji poreikiai žodynams. Reikėjo naujų terminų sąsajos elementams įvardyti, naujų terminų ar bendrinės kalbos žodžių kompiuterio komandoms ir pranešimams. Jais naudojasi ne vien kompiuterijos specialistai, bet ir visi kompiuterių naudotojai. Susipynė informatikos terminija, bendrinės kalbos žodžiai, tos žmogaus veiklos srities, kuriai skiriama programa, leksika ir terminija (Grigas, 2006, 2008). Minėti ir kiti esami žodynai galėjo pasitarnauti tik kaip specialių terminų šaltinis žodynams, skirtiems programinės įrangos lokalizuotojams ir kompiuterio naudotojams.

Vyko diskusijos dėl terminų, buvo stengiamasi juos vienodinti, kad mažiau būtų sinonimų (Kaulakienė, 2000, 2006). Buvo rengiami nauji žodynai. Trikalbį anglų–lietuvių–prancūzų kalbų žodyną ir keletą anglų–lietuvių dvikalbių žodynėlių parengė V. Žalkauskas (2003, 2004a, b, 2005).

Dvikalbiai (ir daugiakalbiai) žodynai tinka sparčiai žodžio vertimo paieškai. Tačiau juose nevisada pakanka informacijos, ypač, kai originalo kalbos žodis nevienareikšmis: ne visada aišku, kurią reikšmę iš žodyne teikiamų reikia imti. Siekiant tiksliai apibrėžti kiekvieną reikšmę ir jos vartojimo sritį, rengiami aiškinamieji ir enciklopediniai žodynai.

„Enciklopedinio kompiuterijos žodyno“ antrajame papildytame leidime (Dagienė ir kt., 2008) enciklopedinė dalis sujungta su toje pat knygoje pateiktu dvikalbiu anglų–lietuvių kalbų žodynu: prie lietuviškų žodžių, esančių dvikalbiame žodyne, pateiktos nuorodos į enciklopedinę dalį (jeigu ten tas žodis aprašytas). Nuorodomis patogu naudotis elektroniniame žodyno variante[1]: radus angliško žodžio

[1] Enciklopedinis kompiuterijos žodynas: http://www.likit.lt/term/enc.html

vertimus galima iš karto per saitą patekti į lietuviško žodžio aprašą ir pasitikslinti, kuri reikšmė geriausiai tinka konkrečiu atveju.

Kai lokalizuojama nauja programa, atsiranda naujų terminų ir šiaip žodžių, kurių nėra bendruose kompiuterijos žodynuose. Todėl pirmiausiai, ypač jeigu programa didesnė, parengiamas tai programai skirtas darbinis žodynas. Į jį įtraukiami toje programoje esantys terminai. Tų, kurie yra bendruose žodynuose, vertimai imami iš tų žodynų, kitiems sugalvojami nauji lietuviški.

Lokalizuotojas neprivalo besąlygiškai laikytis bendruose žodynuose teikiamų terminų. Jeigu gali pasiūlyti geresnį (tikslesnį, aiškesnį, trumpesnį, skambesnį ir pan.), tai jis gali jį vartoti ir siūlyti žodynų autoriams, kad jie įtrauktų į žodynus. Taip lokalizuotojas gali prisidėti prie terminijos kūrimo.

Stambesnės įmonės ar bendrijos viešai pateikia jų programose vartojamą terminiją: žodynus, išteklių eilučių vertimus[1]. Išteklių eilučių vertimais galima naudotis, tačiau į juos reikia žiūrėti kritiškai, nes dažnai būna klaidų, kol kas mažai šie vertimai menkai redaguojami.

7.3. Terminų bazės

Terminų bazė – į daugiakalbį verčiamąjį žodyną su aiškinamaisiais elementais panaši priemonė, tik terminai ir papildoma informacija apie juos laikomi specialioje duomenų bazėje. Terminų bazės gali būti integruotos į kompiuterizuoto vertimo ir lokalizavimo dalinio automatizavimo sistemas kartu su vertimo atmintimis, išteklių tvarkytuvėmis ir kitomis priemonėmis. Gali būti naudojamos atskirai, duomenų bazių valdymo sistemose.

Terminijos duomenys terminų bazėje laikomi įrašų pavidalu: viename įraše laikomas bent vienas terminas. Tipinį terminų bazės įrašą paprastai sudaro tokie laukai: terminas, termino vertimai į vieną ar daugiau kalbų, termino vartojimo sritis, gramatinė informaciją ir kita visą įrašą ir kiekvieną atskirą terminą ar jo komponentą aprašanti informacija.

[1] Išteklių eilučių tekstynai: 1) atviruųjų programų http://open-tran.eu/projects.html; 2) „Microsoft“ firmos http://www.microsoft.com/language/en-us/default.aspx ir kt.

Siekiant užtikrinti terminų bazių mainus tarp įvairių vertimo ir lokalizavimo paslaugų teikėjų, vertimo ir lokalizavimo programinės įrangos, terminų bazių formatai standartizuojami. Šiuo metu žinomas ir paplitęs terminų bazių formatą apibrėžiantis tarptautinis standartas, su kuriuo suderinamos dauguma terminų bazių sistemų, yra TBX. TBX specifikaciją projektuoja Lokalizavimo pramonės standartų asociacija LISA[1], o papildytas jos variantas publikuotas kaip tarptautinis standartas ISO 30042.

TBX standartizuoja terminijos duomenų struktūrizavimą naudojant XML kalbą. TBX standartą sudaro du moduliai, išreikšti XML kalba: pagrindinė struktūra ir duomenų kategorijų identifikavimo formalizmas. Standarte numatytas įvairių terminijos ženklinimo kalbų naudojimas, tačiau standarto numatytoji terminijos ženklinimo kalba yra TML (angl. *Terminological Markup Language*), kurią apibrėžia tarptautinis standartas ISO 16642.

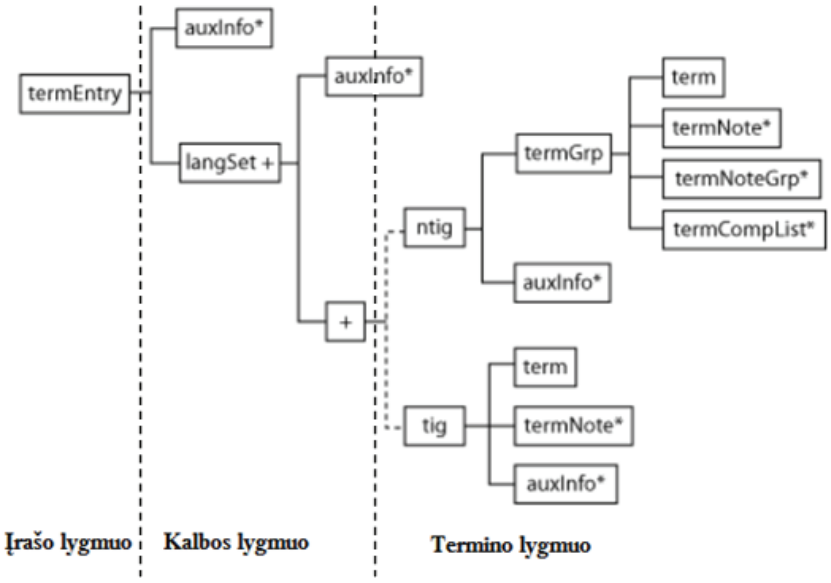
TBX įrašo hierarchinė struktūra (naudojant standartizuotas XML gaires) pavaizduota 51 paveiksle. Jame panaudotų elementų trumpi aprašai pateikti 36 lentelėje.

TBX standarte naudojamos analogiškos įterptinės gairės teksto formatams laikyti, kaip ir vertimo atminčių formate TMX (7.4 skyr.).

TBX standarto svarbiausi taikymo aspektai (ISO 30042):

- Terminų mainai tarp vertėjų (lokalizuotojų), įvairių sistemų ir terminijos sričių.
- Terminijos duomenų integracija iš įvairių šaltinių.
- Duomenų konvertavimas į įvairius formatus.
- Užklausų formavimas daugeliui terminų duomenų bazių naudojant universalią grafinę sąsają ir perduodant duomenis vieningu standartizuotu tarpiniu formatu.
- Terminijos duomenų publikavimas interneto svetainėse.
- Terminų bazės įrašų pildymas kolektyviai, naudojant interneto priemones.
- Terminų bazės dinaminis atnaujinimas internetinėse programose naudojant saityno paslaugas.
- Įvairių terminų bazių įrašų analizė ir palyginimas.
- Duomenų praradimo, konvertavus vieną terminų bazės formatą į kitą, tyrinėjimas.

[1] The Localization Industry Standards Associacion. Term Base eXchange (TBX), http://www.lisa.org/Term-Base-eXchange.32.0.html



51 pav. TBX įrašo lygmenys

36 lentelė. TBX termino įrašą formuojantys pagrindiniai elementai

Elementas	Aprašas
<termEntry>	Terminijos įrašas – konceptas
<auxInfo>	Papildoma informacija
<langSet>	Elementas, kuriame laikoma visa informacija apie terminą tam tikra viena kalba
<ntig>	Informacijos apie terminus struktūrizavimo elementas
<tig>	Elementas, skirtas pateikti visą informaciją apie terminą. Neturi tiek informacijos struktūrizavimo priemonių, kaip elementas <ntig>
<termGrp>	Naudojamas <ntig> elemente skaidant terminą į komponentus
<term>	Terminas. Aprašomas <tig> arba <ntig> elementais
<termNote>	Metaduomenų kategorija, skirta terminams aprašyti. Naudojant šio elmento atributus nusakomas termino tipas
<termNoteGrp>	Pastabų apie terminą grupavimo elementas
<termCompList>	Termino komponentų sąrašo sudarymo ir grupavimo elementas, turintis bent vieną elementą <termComp> arba bent vieną elementą <termCompGrp>, taip pat papildomos informacijos apie terminų komponentus. Šio elemento atributai nurodo, kokio tipo komponentai sudaro terminą

7.4. Vertimo atmintys

Tos pačios programos lokalizuojamuose ištekliuose, dažnai ir skirtingose programose, būna pasikartojančių teksto fragmentų – frazių, sakinių, o kartais ir ištisų pastraipų. Tam, kad jų vertimus būtų galima panaudoti pakartotinai, jie, suporuoti su originalais, sudedami į vertimo atmintį, kuri paprasčiausiu atveju yra tokių eilučių porų rinkinys. Visai kaip žodynas, tik poruojami ne žodžiai, o ilgesni teksto fragmentai, dažniausiai išteklių eilutės.

Ilgos eilutės gali būti skirstomos į dalis, kurios vadinamos teksto segmentais. Skaidymas į segmentus priklauso nuo konkrečios vertimo atminties programinės įrangos. Segmentavimui dažniausiai naudojami tokie skirtukai (Bowker, 2005):

- sakinio pabaigos ženklai,
- kabliataškis,
- kablelis,
- pastraipos pabaigos ženklas,
- sąrašo elemento pabaigos ženklas,
- lentelės langelio pabaigos ženklas.

Vertimo atmintyje įrašyta susieta originalo ir vertimo segmentų pora vadinama vertimo vienetu. Tam tikrose realizacijose vertimo vienetą gali sudaryti nebūtinai segmentų pora, bet originalaus teksto segmentas ir to segmento vertimai į daugiau kaip vieną kalbą. Vertimo vienetai yra paprastai formuojami vertimo metu (išvertus vieną segmentą) arba atlikus visą vertimą ir įdėjus tekstų poras į vertimo atmintį. Taip vertimo atminties duomenų bazė auga lokalizavimo metu.

Vertimo atmintis ypač naudinga atnaujinant lokalizaciją. Naujoje programos laidoje didžioji tekstų dalis lieka ta pati, kaip ir ankstesnėje. Jeigu ankstesnė laida buvo lokalizuota ir į vertimo atmintį sudėti jos išteklių teksto segmentų vertimai, tai atnaujinant lokalizaciją, lokalizuojamos laidos originalo segmentų, kurie sutampa su ankstesnės laidos originalo segmentais, versti nebereikia – jų vertimus galima automatiškai iš ten ir imti. Lieka išversti naujus arba pasikeitusius segmentus ir jais papildyti vertimų atmintį, kad būtų galima ateityje ją panaudoti naujesnei programos laidai lokalizuoti.

Vertimo atmintys gali būti parengtos vienai programai, programų grupei arba universalios visai programinei įrangai. Panašiai, kaip ir žodynai.

Svarbi vertimo atminties dalis yra paieška, kuri būna dviejų tipų:

- tikslioji;
- apytikslė.

Kai vertimo atmintyje randamas vertimo vienetas, visiškai sutampantis su ieškomu, tai viskas aišku ir paprasta. Tačiau ne visada taip būna, ypač kai atmintis panaudojama kitai programai lokalizuoti. Segmento prasmė gali būti ta pati, bet šiek tiek skirtis jo tekstas. Gali būti nevienodai pavartoti pagalbiniai žodžiai, sinonimai. Pavyzdžiui, frazių „Applications should be sent to“ ir „Please forward your applications to“ semantika yra panaši, ir abu šie segmentai į kitą kalbą galėtų būti verčiami ta pačia fraze, tačiau paieškos vertimo atmintyje metu nebūtų rastas kitaip suformuluoto originalaus teksto vertimas.

Kita situacija – klaidingas tikslaus atitikmens radimas, kai vartojama kita žodžio ar frazės prasmė (pvz., „portrait“ – portretas, stačias (lapas); „position“ – padėtis, pareigos).

Todėl kartu su teksto segmentu kai kuriose priemonėse gali būti pateikiama papildoma informacija apie jame esančius žodžius: kurie pagrindiniai, kurie pagalbiniai. Atskirai terminų bazėje laikoma informacija apie atskirus žodžius: kurie yra jo sinonimai, kokiame tekste vienas ar kitas žodis geriau tinka ir pan.

Jeigu nerandama tikslios atitikties, tai paieškos mechanizmas, pasinaudodamas šia informacija, gali pateikti apytikslės paieškos rezultatą, nurodydamas ir atitikties laipsnį, paprastai išreiškiamą procentais. Tada lokalizuotojas turi patikrinti, ar toks vertimas tinkamas, ir gali jį priimti, koreguoti arba pateikti naują.

Apytikslės atitikties panašumas vadinamas atitikmens kokybe. Atitikmens kokybės matavimo algoritmai skiriasi įvairiose vertimo atminčių sistemose (pvz., segmentų panašumui nustatyti seniai žinomas Levenšteino atstumo algoritmas, kuriuo galima rasti skaičių įterpimų, šalinimų ir pakeitimų, reikalingų vieną frazę paversti kita).

Vertimo atmintis gali būti sujungta su terminų bazėmis ir dažniausiai yra vienas svarbiausių kompiuterizuoto vertimo priemonių komponentų. Tada mažiau darbo liktų lokalizuotojui: jis galėtų daugiau dėmesio skirti naujiems tekstams lokalizuoti, išlaikydamas terminijos suderinamumą su ankstesnių versijų ar panašios paskirties programinės įrangos vertimais.

Vertimo atminties programa paprastai teikia įvairią statistinę informaciją: žodžių, segmentų skaičių, tikslių ir apytikslių eilučių, klasifikuotų pagal atitikties laipsnį, naujų segmentų skaičių.

7.4.1. Vertimo atminčių mainų formatas TMX

Vertimo atminčių idėja atsirado šiek tiek anksčiau, negu prasidėjo realus programinės įrangos lokalizavimas. Kai kurie autoriai (pvz., Melby, Warner, 1995) nurodo, kad vertimo atmintis buvo sumanyta XX a. aštuntajame dešimtmetyje, tačiau pradėta realizuoti tik pačioje XX a. pabaigoje. Vertimo atminčių programų kūrėjai sugalvodavo savus atminčių formatus, savas tekstų segmentavimo taisykles. Todėl parengus vertimo atmintį naudojantis viena priemone, jos nebuvo galima naudoti lokalizuotojams, dirbantiems su kita vertimo atminčių programa. Atsirado vertimo atminčių formatų mainų ir standartizavimo klausimas.

Pagrindinė šiuo metu žinoma ir plačiai naudojama įvairių lokalizavimo priemonių kūrėjų vertimo atminčių vieningo formato specifikacija yra TMX.

TMX formatą projektuoja LISA asociacijos standartų komitetas OSCAR[1]. Tai nuo programinės įrangos gamintojo nepriklausomas formatas, skirtas vertimo atminties duomenų, sukurtų kompiuterizuoto vertimo ir lokalizavimo priemonių, mainams. TMX formato paskirtis – palengvinti vertimo atminčių duomenų tarp įvairių lokalizavimo (vertimo) priemonių ir lokalizavimo proceso dalyvių mainus, neprarandant svarbių duomenų arba sumažinant tokių duomenų praradimą iki minimumo.

TMX formatas yra atvirasis ir kuriamas XML kalbos pagrindu[2]. TMX dokumento elementus galima suskirstyti į dvi grupes:

- struktūriniai elementai,
- įterptiniai elementai.

Pagrindinis funkcinis struktūrinis elementas yra vertimo vienetas, kurį atitinka gairė `<tu>`. Kiekvienas vertimo vieneto elementas turi identifikatorių. TMX vertimo vieneto elementas gali laikyti bent dviejų kalbų tekstų fragmentus. Tam specifikacijoje yra vertimo vieneto varianto `<tuv>` elementas. Verčiamų tekstų segmentų pradžią ir pabaigą žymi segmento gairė `<seg>` (52 pav.).

Papildomą informaciją į TMX failą galima įrašyti dviem pagrindiniais būdais:

1. Komentarų pavidalu, panašiai, kaip ir kituose lokalizuojamųjų išteklių formatuose (5.3 skyr.). Tam naudojama TMX gairė `<note>`, pavyzdžiui, `<note>Ši eilutė buvo sutrumpinta tam, kad tilptų dialogo lange.</note>`.

[1] OSCAR, LISA's Standards Committee, http://www.lisa.org/OSCAR-LISA-s-Standards-Committee.79.0.html

[2] TMX specifikaciją galima rasti LISA svetainėje www.lisa.org/tmx.

```
...
<tu tuid="18" datatype="Text" srclang="en-us">
   <prop type="x-Project">8526270</prop>
   <tuv xml:lang="en-us">
     <seg>Bookmarks</seg>
   </tuv>
   <tuv xml:lang="lt">
     <seg>Adresynas</seg>
   </tuv>
</tu>
<tu tuid="19" datatype="Text" srclang="en-us">
   <prop type="x-Project">8526270</prop>
   <tuv xml:lang="en-us">
     <seg>Create in:</seg>
   </tuv>
   <tuv xml:lang="lt">
     <seg>Įrašyti į aplanką:</seg>
   </tuv>
</tu>
<tu tuid="20" datatype="Text" srclang="en-us">
   <prop type="x-Project">8526270</prop>
   <tuv xml:lang="en-us">
     <seg>Text in <bpt i="1">&lt;b></bpt>bold<ept i="1">&lt;/b></ept>.</seg>
   </tuv>
   <tuv xml:lang="lt">
     <seg>Tekstas <bpt i="1">&lt;b></bpt>pusjuodžiu<ept i="1">&lt;/b></ept> šriftu.</seg>
   </tuv>
  </tu>
...
```

52 pav. TMX failo fragmentas: dviejų vertimo vienetų pavyzdys

2. Nurodant vertimo vieneto savybes (pvz., lokalizavimo projekto numerį, sritį, failų tipus) naudojama <prop> elementas. Pavyzdžiui, lokalizuojamos programos išteklių terminijos sritį ir stilių galima nurodyti taip: <prop type= "x-Domain">Programinė įranga pradinių klasių mokiniams</prop>.

Įterptiniai elementai naudojami vertimo vienetų tekstų formatavimo informacijai, ateinančiai iš įvairių lokalizuojamųjų išteklių ar dokumentų formatų, išsaugoti, kad būtų galima ją identišką grąžinti į originalius formatus (37 lentelė).

Įterptinės gairės priklauso nuo originalaus dokumento ar lokalizuojamųjų išteklių formato. HTML dokumente pusjuodžio šrifto formatą turintis tekstas (porinės gairės ...) TMX dokumente būtų pažymėtas gairių <bpt>...<ept> pora. RTF formate pusjuodžiu šriftu formatuotas tekstas prasideda kodu „\b“, o baigiamas gali būti „\b0“, todėl teisinga TMX įterptinė gairė tokiam atvejui naudoti yra neporinė gairė <ph>. Vertimo priemonės, suderinamos su TMX formatu, gali įkelti vertimus iš įvairių pradinių formatų, nepaisant formatavimo skirtumų.

TMX failo fragmentas, kuriame vertimo vieneto formatavimas pažymėtas TMX įterptinėmis porinėmis gairėmis <bpt> ir <ept>, pavaizduotas 53 paveiksle.

37 lentelė. TMX įterptiniai elementai

Elementas	Aprašas
<bpt>	Porinės gairės pradžia. Naudojama atskirti originalaus formato porinių formatavimo gairių žymimą sekos pradžią. Kiekvienas segmente esantis <bpt> elementas turi atitinkamą <ept> elementą
<ept>	Porinės gairės pabaiga. Naudojama atskirti originalaus formato porinių formatavimo gairių žymimą sekos pabaigą. Kiekvienas segmente esantis <ept> elementas turi atitinkamą <bpt> elementą
<hi>	Paryškinimas. Atskiria teksto bloką, turintį specialią prasmę, pavyzdžiui, terminijos vienetą, registruotą vardą ar elementą, kuris neturėtų būti verčiamas. Naudojamas įvairiai: vertimo ir lokalizavimo priemonei nurodyti, kurių teksto dalių nepateikti vertimui; terminijai verifikuoti; patikrinus gramatiką, gali žymėti klaidingus ar įtartinus tekstus
<it>	Izoliuota gairė. Naudojama atskirti originalaus formato formatavimo gairių žymimą sekos pradžią (pabaigą), kurių atitinkama pabaiga (pradžia) yra kitame segmente
<ph>	Neporinė gairė. Naudojama atskirti originalaus formato neporinės gairės žymimą seką segmente
<sub>	Vidinė gairė. Naudojama atskirti tekstą originalaus formato kodo sekos viduje, pavyzdžiui, išnašos žymėjimą ar HTML žymės elemento pavadinimą
<ut>	Nežinoma gairė. Naudojama atskirti originalaus formato nežinomo kodo seką

TMX specifikacija nenurodo tekstų segmentavimo taisyklių. Su specialiais XSL stiliais galima pateikti skaitymui apipavidalintą vertimo atmintį ar jos dalį tinklalapyje (Musale, 2004).

TMX specifikacija apibrėžia du lygius vertimo atminčių mainų:

1. Grynojo teksto. Į TMX dokumentą įtraukiamas tik vertimui skirtas grynasis tekstas, atmetus visą formatavimo informaciją. Tai reiškia, kad segmentuose (<seg> elementuose) nenaudojamos įterptinės gairės.
2. Raiškiojo teksto. Į TMX dokumentą įtraukiamas vertimui skirtas tekstas su jo formatavimo informacija (jei tokia yra). Naudojamos įterptinės gairės.

```
...
<tu tuid="20" datatype="Text" srclang="en-us">
   <prop type="x-Project">8526270</prop>
   <tuv xml:lang="en-us">
     <seg>Text in <bpt i="1">&lt;b></bpt>bold<ept i="1">&lt;/b></ept>.</seg>
   </tuv>
   <tuv xml:lang="lt">
     <seg>Tekstas <bpt i="1">&lt;b></bpt>pusjuodžiu<ept i="1">&lt;/b></ept> šriftu.</seg>
   </tuv>
  </tu>
...
```

53 pav. TMX failo fragmentas: vertimo vienetas su įterptiniais elementais

Jei nurodoma, kad vertimo atminčių priemonė yra suderinama su TMX standartu, turi būti patikslinta, kuris suderinamumo lygis yra užtikrintas. Iki 2005 metų įvairių gamintojų lokalizavimo priemonėse dominavo pirmojo lygio suderinamumas (Zerfass, 2005). Pastaraisiais metais realizuojamas ir antrojo lygio suderinamumas. Tačiau vertimo atminčių antrojo lygio mainų srityje dar esama problemų: kai kurios priemonės naudoja senesnes TMX specifikacijos versijas, kitos nenaudoja XML validavimo priemonių. Šias problemas tikimasi išspręsti šiuo metu projektuojamoje TMX specifikacijos 2.0 versijoje. Joje planuojama specifikuoti įterptines gaires, suderinamas su XLIFF standarto atitinkamomis gairėmis (5.2 skyr.).

7.4.2. Lokalizavimo produktyvumo ir kokybės santykis naudojant vertimo atmintis

Pradėjus taikyti vertimo atmintis lokalizavimo procese, pradėtas tirti lokalizavimo (vertimo) produktyvumo ir kokybės santykis (pvz., Bowker, 2005; Arenas, 2008).

Vertimo atminties taikymo efektyvumas priklauso nuo šių veiksnių:

- naudojamos vertimo atminties duomenų bazės dalykinės srities;
- vertimo atmintyje esančių segmentų vertimų kokybės;
- vertimo atmintyje esančių segmentų kiekio: efektyvumas turėtų augti pildantis vertimo atminčiai;
- panašių verčiamų tekstų kiekio;
- vertėjų, dirbančių su vertimo atmintimi, kvalifikacijos.

Remiantis Somers atliktais tyrimais (2003), vertėjo, dirbančio su vertimo atmintimi, darbo produktyvumas padidėja vidutiniškai 30 proc. Kai kurie autoriai laiko, kad darbo produktyvumas padidėja net 50 ar 70 proc. Tačiau Bowker (2005) tyrimai rodo, kad darbo su vertimo atmintimi pradžioje darbo produktyvumas mažėja, o vertimų klaidų skaičius yra didesnis dirbant su vertimo atmintimi, negu be jos. Remiantis Arenas (2008) atliktu tyrimu, vertimo atminties taikymas gali mažinti vertimo kokybę ir net darbo produktyvumą, kai vertimo atmintyje yra nuo 80 iki 90 proc. apytikslių atitikmenų.

Dažnai pažymima, kad vertimo atminties taikymas lokalizavimo (vertimo) procese padeda išlaikyti vertimų vienodumą, suderinti terminiją (Bowker, 2005). Tai, be abejo, svarbus privalumas, tačiau esant tam tikroms sąlygoms, jis gali būti trūkumu. Vertimo atmintyje laikomi segmentai yra tekstai, sakiniai ar frazės be konteksto.

Jie gali patekti į kitą kontekstą, kuriame iš tikrųjų reikėtų kitokio vertimo. Bowker taip pat pataria neįtraukti visų verčiamų tekstų ar lokalizuojamų eilučių į vertimo atmintį. Mažesnė vertimo atmintis su kruopščiai atrinktais segmentais yra vertingesnė už didesnę vertimo atmintį, į kurią segmentai buvo įtraukiami chaotiškai.

7.4.3. Vertimo atminčių naudojimo pagrindiniai privalumai ir trūkumai

Apibendrinant įvairių tyrėjų nuomones (Esselink, 2000; Arenas, 2008; Musale, 2004; Bowker, 2005), galima išskirti tokius vertimo atminčių naudojimo lokalizavimo procese privalumus:

- Į naujus vertimus galima įtraukti ankstesnių vertimų fragmentus. Programinės įrangos elektroniniuose žinynuose ir dokumentacijoje gausu pasikartojančių tekstų, kuriuos galima panaudoti ne tik lokalizuojant tos pačios programos naują versiją, bet ir kitas programas.
- Lokalizavimą galima pradėti dar iki išleidžiant naują programos versiją. Galima pasinaudoti vertimo atminties pasiūlytais apytiksliais atitikmenimis. Taip paspartinamas naujų lokalizuotų versijų išleidimas.
- Vertimas paspartėja (jei vertimo atmintis nėra tuščia ir joje yra panašių teksto segmentų).
- Vertimus galima kaupti, analizuoti, vienodinti terminiją. Vertimo atmintis yra tarsi žodynas ar frazynas.
- Vertimo atmintimis galima dalytis. Dalijamasi vertimo atmintimis ne tik tam tikros organizacijos viduje. Paplitus interneto priemonėms, pradėtos naudoti centralizuotos vertimo atmintys, esančios tinkle, prie kurių gali prisijungti vertėjai ir lokalizuotojai iš viso pasaulio, pasinaudoti didžiulėmis pasaulinėmis vertimo vienetų saugyklomis bei nuolat pildyti jas.

Pagrindiniai vertimo atminčių trūkumai, apibendrinant įvairių tyrėjų duomenis, yra tokie:

- Pradėjus taikyti vertimo atmintis neišvengiamai turi būti pakeistas lokalizavimo ir vertimo proceso organizavimas.
- Reikalingas specialus lokalizuotojų pasiruošimas.
- Pramonėje naudojamos vertimo atminčių priemonės yra brangios. Atvirųjų priemonių nėra daug, jos turi mažiau funkcijų, negu komercinės.

- Kai tekstas automatiškai segmentuojamas, lokalizuotojas negali pakeisti verčiamo teksto segmentų išdėstymo eilės.
- Yra pavojus mechaniškai priimti atminties pasiūlytą vertimą, netinkamą konkrečiam kontekstui.
- Nepastebėtas klaidingas vertimas, esantis vertimo atmintyje, gali patekti į naujų segmentų vertimus.
- Įvairių vertėjų sukurtoje vertimo atmintyje gali būti skirtingo stiliaus ar net prieštaringų vertimų.
- Nepakankamai lankstus tekstų formatų, esančių vertimo atminties segmentuose, laikymas ir mainai.
- Lokalizuojamos programinės įrangos kompleksinio testavimo metu aptiktos lokalizavimo klaidos dažnai taisomos tiesiai lokalizuojamuose ištekliuose, pamirštant tuos taisymus įkelti į vertimo atmintį.

7.5. Lokalizuojamųjų išteklių tvarkytuvės

Lokalizuojamųjų išteklių tvarkytuvės skirtos įvairių formatų (5.2 skyr.) ištekliams lokalizuoti juos vizualizavus.

Išteklių tvarkytuvės gali būti įvairiai klasifikuojamos, pavyzdžiui, pagal:

- veikimo būdą:
  - klientinės (veikiančios klientų kompiuteriuose);
  - tinklinės (veikiančios tinklo serveryje);
- apdorojamų failų formatų įvairovę:
  - specializuotos tam tikro išteklių atskyrimo metodo formatų išteklių tvarkytuvės;
  - daugelio išteklių formatų rengyklės;
- išteklių failų tekstų ilgius:
  - trumpų tekstų (pvz., grafinės sąsajos tekstų) išteklių tvarkytuvės;
  - ilgų (rišlių) tekstų išteklių tvarkytuvės.

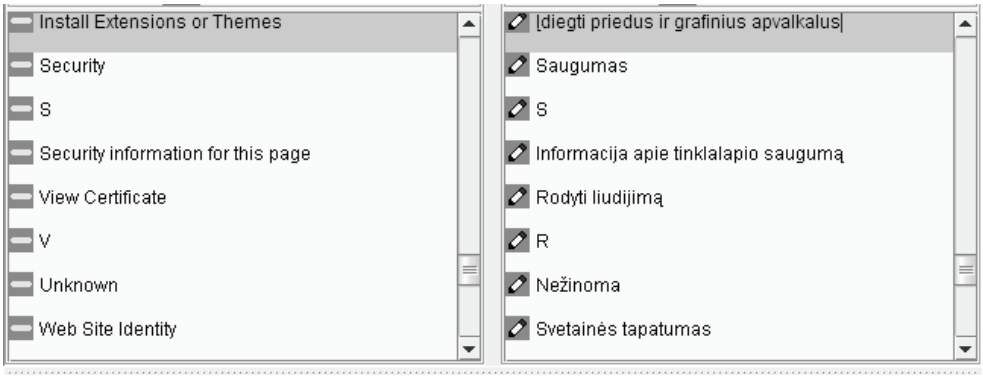
Klientinės išteklių tvarkytuvės skirtos autonomiškai dirbantiems lokalizuotojams. Tinklinės veikia tinklo serveryje, dažniausiai pasiekiamos naršykle. Išteklių failai tokiu atveju gali būti laikomi serveryje, o su jais gali dirbti lokalizuotojų komanda.

Dalį 5.2 skyriuje nagrinėtų lokalizuojamųjų išteklių formatų (pvz., „Gettext PO“, PROPERTIES, DTD, XLIFF, PHP ir kt.) galima versti naudojant grynojo teksto rengykles. Patogiau dirbti su specialiomis lokalizavimui skirtomis išteklių tvarkytuvėmis. Jos dažniausiai nerodo dėmesį blaškančios tarnybinės formatų informacijos, o pateikia tik tai, kas lokalizavimui svarbu, suskaido tekstą į vertimui skirtus segmentus ir dažniausiai parodo originalo tekstą ir vertimą greta (54 pav.). Tuo skiriasi nuo bendrųjų atitinkamų formatų dokumentų rengyklių.

Išteklių tvarkytuvių funkcijos įvairiuoja priklausomai nuo lokalizuojamųjų išteklių formato, kurį jos gali apdoroti. Pavyzdžiui, su RC ar RESX lokalizuojamaisiais ištekliais dirbti skirtos rengyklės pateikia ne tik verčiamus programos pranešimus ir kitus tekstus, skirtus rodyti ekrane, bet ir statinius programos dialogo langus (tam naudojama formato scenarijaus dialogo sekcija, žr. 5.2 skyrių), kuriuose galima interaktyviai keisti langų, mygtukų, įvairių laukų dydžius. Tačiau jos apdoroja tik statinius langus. Jomis neįmanoma sumodeliuoti visų programos vykdymo metu atsirandančių situacijų ir pranešimų. Tokios rengyklės taip pat leidžia pakeisti (lokalizuoti) į EXE ir DLL failus įkompiliuotas piktogramas ir paveikslus.

Jei lokalizuojamųjų išteklių formate numatytas lokalizuojamųjų eilučių komentavimas, žymėjimas, tai tokių išteklių tvarkytuvės paprastai leidžia lokalizuotojui (vertėjui) interaktyviai įterpti komentarus ar žymes, pavyzdžiui, ar vertimas baigtas ir patikrintas, ar vertimas tikslintinas ir pan.

Kai kurios išteklių tvarkytuvės skirtos dirbti su keliais lokalizuojamųjų išteklių formatais, pavyzdžiui, DTD, PROPERTIES, PO, XLIFF. Į daugelį išteklių tvarkytuvių paprastai būna integruojami išteklių konvertavimo iš vieno formato į kitą filtrai.



54 pav. Tekstinių programos išteklių vertimas išteklių tvarkytuvėje

Panašiai, kaip vertimo atmintyse (7.4 skyr.), išteklių tvarkytuvėse ilgesni tekstai gali būti skaidomi į segmentus. Trumpi grafinės sąsajos tekstai ir pranešimai (žodis, frazė ar neilgas sakinys) dažniausiai nesegmentuojami: viena eilutė atitinka vieną segmentą.

Į lokalizavimo priemonių paketus dažniausiai įtraukiamos ir vertimo atmintys, ir išteklių tvarkytuvė. Originalus segmentas ir jo vertimo pora iš išteklių tvarkytuvės siunčiama į vertimo atmintį.

Siekiant suvienodinti segmentavimo taisykles ir gerinti vertimo atminčių, išteklių tvarkytuvių ir kitų tekstų vertimo ir apdorojimo priemonių suderinamumą, teksto segmentavimo taisyklės standartizuojamos. Tokį standartą[1] yra parengusi LISA asociacija.

Kai kurios išteklių tvarkytuvės turi pseudovertimo funkciją. Siekiant įsitikinti, kad visos kalbos abėcėlės raidės bus matomos teisingos, verstinos eilutės dažniausiai pakeičiamos lokalizuojamos kalbos raidžių, kurių nėra anglų kalbos abėcėlėje, rinkiniais. Taip pat preliminariai patikrinama, ar tekstams rodyti ekrane skirtose vietose išsiteks ilgesnės, negu originalo kalbos, eilutės. Visa tai atliekama dar prieš tikrąjį lokalizavimą.

Lokalizuojamos programinės įrangos žinyno ir dokumentacijos failai būna įvairių formatų. Dalį jų galima lokalizuoti naudojantis tomis pačiomis išteklių tvarkytuvėmis, kuriomis lokalizuojamos sąsajos eilutės. Tokiu atveju ilgesni žinyno tekstai segmentuojami ir verčiami analogiškai sąsajos eilutėms. Yra žinynų, kurių galutinis formatas skiriasi nuo jų rengimo formato. Pavyzdžiui, „Windows“ sistemose vartojami HLP žinynų failai, kurie gaunami iš RTF formato.

7.6. Kompiuterinio vertimo, lokalizavimo apskaitos ir testavimo priemonės

Kompiuterinio vertimo priemonės. Tai programos, skirtos tekstams automatiškai versti iš vienos kalbos į kitą. Jos negali pakeisti žmogaus, tačiau jos gali tikti juodraštiniams vertimams parengti.

Natūraliosios kalbos turi labai turtingas raiškos formas. Todėl sunku viską iš anksto numatyti, juo labiau – formalizuoti ir automatizuoti. Dėl to kol kas gaunamas

[1] SRX (angl. *Segmentation Rules eXchange*), http://www.lisa.org/Segmentation-Rules-e.40.0.html

žemos kokybės vertimas, kuris gali būti panaudotas tik kaip ruošinys vertėjui. Grafinės sąsajos tekstų, kurie pasižymi lakoniškumu ir kurių kontekstas programoje nėra žinomas (5.3 skyr.), kompiuterinis vertimas dar labiau problematiškas. Automatinio vertimo į lietuvių kalbą sistemos tebėra projektavimo stadijoje.

Kompiuterinio vertimo priemonės naudojamos autonomiškai arba integruotos į lokalizavimo priemonių paketus. Pastaruoju atveju segmento vertimo metu ieškoma atitikmens vertimo atmintyje, terminų bazėje, o jei nerandama, siūlomas kompiuterinio vertimo priemonės generuotas vertimas.

Lokalizavimo apskaitos priemonės. Prieš pradedant lokalizuoti programą, reikia numatyti būsimų darbų apimtį ir sąnaudas. Todėl reikalingos priemonės, kurios padėtų įvertinti lokalizuotinų eilučių, žodžių, grafinių failų ir kitų išteklių kiekius.

Lokalizavimo apskaitos priemonės leidžia atlikti dviejų pagrindinių rūšių analizę.

- Lokalizuojamųjų išteklių statistinė analizė. Skaičiuojami žodžiai, eilutės ir jų ilgiai, puslapiai, įvairialypės terpės ir kiti lokalizuotini elementai. Pateikus atskirų lokalizavimo darbų įkainius galima apytikriai apskaičiuoti programos ar svetainės lokalizavimo kainą.
- Darbo laiko, reikalingo kuriam nors lokalizavimo darbui atlikti, analizė. Lokalizavimo (vertimo) veiksmai ir jų atlikimo eiga įrašomi į specialų žurnalą. Naudojantis šia informacija galima analizuoti įvairių lokalizavimo darbų laiko sąnaudas, planuoti lokalizavimo darbams reikalingą laiką, tirti lokalizavimo komandos darbo efektyvumą.

Lokalizavimo testavimo priemonės. Padeda aptikti klaidas, padarytas lokalizavimo metu arba atsiradusias dėl nepakankamo originalios programos internacionalizacijos lygio. Automatinės testavimo priemonės atlieka tik kai kuriuos testavimo veiksmus. Pagrindinį testavimo darbą vis tik atlieka žmogus.

Testavimo priemonės paspartina tuos programos ar svetainės lokalizavimo veiksmus, kuriuos techniškai galima automatizuoti:

- Patikrina lokalizuotos svetainės ar žinyno saitų (tinklalapių, paveikslų, garso failų ir pan.) veikimą. Tikrinami lokalizavimo metu pridėti ir pašalinti saitai (t. y. ar lokalizuojant per klaidą nepakeisti saitų adresai).
- Patikrina lokalizuotos svetainės ar žinyno hiperteksto gairių dermę.
- Patikrina, ar nėra to paties lygio meniu ar to paties dialogo lango fragmento pasikartojančių prieigos klavišų.
- Patikrina, ar nedubliuojami lokalizuotos programos komandų klavišai.
- Pažymi programos grafinės sąsajos vietas, kur lokalizuoti tekstai netilpo į jiems skirtą vietą.

Įvairių kompanijų ir atviruųjų programų bendruomenių lokalizavimo praktikoje taip pat naudojamos testavimo priemonės, kurios nėra susijusios su testavimo veiksmų automatizavimu, tačiau padeda kaupti iš įvairių testuotojų gautus testavimo rezultatus, juos pateikti ir lyginti. Tokios priemonės remiasi iš anksto sudarytais klausimynais. Į tokių klausimynų klausimus reikia atsakyti atliekant testavimo veiksmus rankiniu būdu. Atsakymai sužymimi ir jų įvertinimai įrašomi į duomenų bazę. Tokias testavimo duomenų bazes tikslinga kurti tada, kai dažnai išleidžiamos lokalizuojamos programos versijos, lokalizuojama daugeliui kalbų.

Kita su testavimu susijusi priemonių rūšis – klaidų registravimo sistemos. Tokios sistemos paprastai būna skirtos ir projektavimui, ir lokalizavimui. Jose gali būti registruojamos iš anksto nenuspėjamos klaidos.

7.7. Kitos priemonės

Tai įvairios priemonės, kurių pagrindinė paskirtis nėra lokalizavimas, bet gali būti panaudotos lokalizavimo darbams.

Hiperteksto rengyklė. Daugelio programų dokumentacija (žinynai) pateikiami hiperteksto pavidalu (pvz., HTML ar XHTML formatais). Lokalizuojant programą dažnai nepakanka vien tik išversti žinyną, reikia jį papildyti, o kartais ir parašyti naują (būna atvejų, kai originalioje programoje žinyno nėra, jis nėra išsamus, nepritaikytas galutiniam programos naudotojui). Tokiais atvejais (dokumentacijai parengti ar iš esmės pertvarkyti) anksčiau aprašytų lokalizavimo priemonių gali nepakakti. Todėl lokalizavimo procese dažnai reikia pasitelkti hiperteksto rengykles (šios programos taip pat tiktų lokalizuojant svetaines ir kitą saityne skirtą publikuoti turinį). Kita hiperteksto rengyklių panaudojimo sritis – lokalizuotų programų lietuviškų svetainių kūrimas.

Šešioliktainė rašyklė. Dirbant su lokalizuojamaisiais ištekliais gali prireikti patikrinti lokalizuojamųjų išteklių koduotę, nustatyti teksto eilučių pabaigos kodus („Windows“, „Linux“ ir „Mac OS“ sistemose jie skirtingi), manipuliuoti tiesiogiai ženklų kodais. Tam gali pasitarnauti šešioliktainė rašyklė arba žiūryklė. Paprastai greta būna rodomi šešioliktainiai kodai ir normalus tekstas. Galima rašyti ir taisyti pačius ženklus (55 pav., dešinėje) arba jų kodus.

Šešioliktainė rašyklė taip pat naudojama dvejetainiams failams taisyti.

```
54 65 6B 73 74 6F 20 70|61 76 79 7A 64 79 73 2C | Teksto pavyzdys,
20 6D 61 74 6F 6D 61 73|20 F0 65 F0 69 6F 6C 69 |  matomas šešioli
6B 74 61 69 6E EB 73 20|72 61 F0 79 6B 6C EB 73 | ktainės rašyklės
20 6C 61 6E 67 65 20   |                        |  lange
```

55 pav. Teksto pavyzdys, matomas šešioliktainės rašyklės lange

Grafikos rengyklės. Programinės įrangos lokalizavimas apima ne tik programos tekstų ir pranešimų vertimą, bet įvairios grafikos, panaudotos programoje, lokalizavimą, pritaikant ją lokalei. Jei originali grafika neatitinka lokalės normų (žr. 8.3 ir 9 skyr.), tenka kurti naujus grafinius failus arba koreguoti esamus. Todėl lokalizuotojo priemonių arsenale turėtų būti taškinės ir vektorinės grafikos rengyklės.

Daug grafikos naudojama programos dokumentacijoje ir elektroniniame žinyne. Čia būna gausu programos langų ar jų fragmentų paveikslų. Lokalizavimo metu būtina visus tokius paveikslus pakeisti, kad atitiktų lokalizuotą programos sąsają, visi užrašai juose būtų tokie, kaip lokalizuotoje programoje. Tam reikalinga ekrano fragmentų iškirpimo ir įrašymo į failus programinė įranga, tai pat grafikos rengyklės.

Modifikuoti arba kurti naują grafiką tenka kuriant lokalizuotos programos svetainę. Jei originaliame paveiksle yra tekstas, tai internacionalizuotos programos ar svetainės grafinių failų ištekliuose turėtų būti naudojami sluoksniai. Taigi tekstui grafikoje lokalizuoti reikėtų sluoksnių tvarkymo funkciją turinčios grafikos rengyklės.

Animacijos, vaizdo ir garso rengyklės ir animacijos lokalizavimo priemonės. Šių priemonių gali prireikti lokalizuojant:

- programų žinynus ir interaktyvius naudotojo vadovus, kuriuose aiškinama, kaip naudojantis programa tvarkyti vaizdus ar animaciją;
- svetaines;
- programinę įrangą, kurioje naudojamas įgarsinimas (pvz., mokomąją programinę įrangą).

Visus animacijos, vaizdo, garso failus, kaip ir statinius grafinius failus, būtina lokalizuoti. Pasitelkiamos bendrosios animacijos, vaizdo rengyklės, garso įrašymo ir tvarkymo priemonės. Rinkoje esama ir specializuotų priemonių, kurios leidžia ištraukti tekstą iš „Flash“ animacijos failų ir jį lokalizuoti.

Programavimo ir kompiliavimo priemonės. Lokalizuojant daugelį programų reikia atlikti programavimo, platinamųjų paketų kompiliavimo darbus. Kadangi lokalizuojamos programos būna nepakankamai internacionalizuotos, tenka lokalę neatitinkančias programų savybes programuoti iš naujo, keisti programos grafinės

sąsajos elementus (jei leidžia lokalizuojamos programos licencija). Todėl būtinos atitinkamos programavimo priemonės. Konkreti programavimo priemonė priklauso nuo lokalizuojamai programai projektuoti naudojamos programavimo kalbos ir programavimo paketo.

Programavimo priemonių naudojimas priklauso nuo konkrečios programos lokalizavimo aplinkybių, ryšių su autoriais. Lokalizuojant komercines programas programavimo, lokalizacijų kompiliavimo, platinamųjų paketų parengimo, internacionalizavimo klaidų taisymo ir kitus darbus paprastai atlieka programos autoriai. Lokalizuojant atvirąją arba nemokamą programinę įrangą dalį programavimo darbų, susijusių su lokalizavimu ir adaptavimu, dažnai tenka atlikti lokalizuotojui.

8. PROGRAMAVIMO KALBŲ IR SVETAINIŲ LOKALIZAVIMAS

Šiame skyriuje aptarsime trijų tipų programinės įrangos priemonių lokalizavimą: programavimo kalbų, kompiliatorių ir svetainių. Pirmieji du tipai labiau specifiniai ir reikalingi siauresniam specialistų ratui – ne kiekvienas lokalizuotojas programuoja ar kompiliuoja programas. Trečiasis tipas – svetainių lokalizavimas – labai plati tema, išeinanti už šios monografijos tikslų: mat reikėtų kalbėti ir apie svetainėms kurti skirtų programinių priemonių lokalizavimą, ir apie jų valdymo adaptavimą kurios nors šalies rinkai, ir apie globalių svetainių dalinį pritaikymą kuriai nors šaliai, ir apie vietinių svetainių – tinklalapių – adaptavimą. Be to, ir pačių svetainių yra įvairiausių, reikėtų jas klasifikuoti, sisteminti, išskirti bendras tendencijas. Nedaug tėra mokslinių publikacijų svetainių lokalizavimo tema, dauguma jų susietos su elektroniniu verslu.

Programavimo kalbų lokalizavimas – gana problematiškas klausimas. Mat vyrauja požiūris, kad programuotojai dažniausiai dirba „be sienų“, globalioje rinkoje ir jie noriau naudojasi originaliomis programavimo kalbų versijomis ir kompiliatoriais (o jie paprastai – anglų kalba). Tačiau reikėtų nepamiršti pradedančiųjų programuotojų rengimo, ypač mokyklinio amžiaus, – jiems lokalizuotos programavimo kalbos, ypač kompiliatoriai, padėtų greičiau perprasti esmines konstrukcijas.

Iš kitos pusės, programavimo kalba – tai ne tik bazinių žodžių ar standartinių procedūrų rinkinys (dėl jų vardų vertimo tikslingumo galima ir abejoti, dauguma komandų tapusios tarptautinėmis ir nesunkiai suprantamos daugumoje terpių), bet ir daugybė papildomos informacijos, kuri turi būti pritaikyta šalies bendruomenei.

8.1. Programavimo kalbos

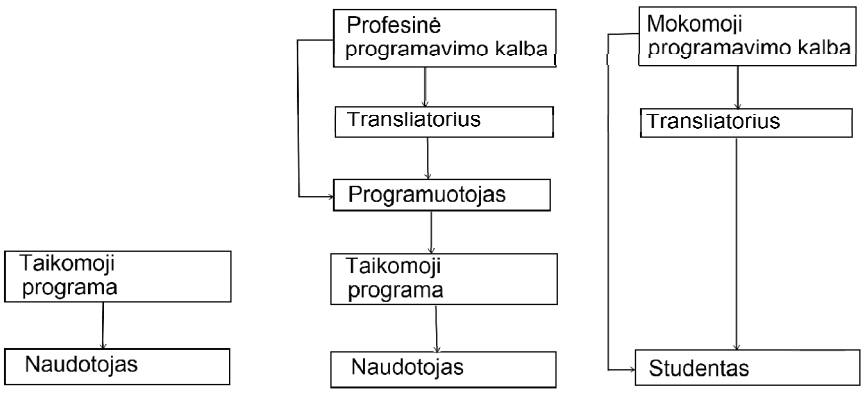
Programavimo kalba yra priemonė programų tekstams rašyti, t. y. programuoti. Programavimo kalbos lokalizavimas – tai adaptavimas vienos ar kitos teritorijos (šalies) kalbos ir kultūros terpei, būtent, tai šaliai, kurioje dirba žmogus, tiksliau – programuotojas.

Lokalizuojant programavimo kalbą siekiama, kad, pirmiausia, būtų patogu ja naudotis pačiam programuotojui, ypač – pradedančiajam, antra, programavimo kalba parašytos originalios taikomosios programos atitiktų tos šalies kalbos ir kultūros terpę ir, trečia, programavimo kalba tiktų internacionalizuotoms taikomosioms programoms rašyti.

Pirmasis reikalavimas apibūdina santykį tarp kalbos ir naudotojo. Tai analogiška santykiui tarp lokalizuotos programos ir jos naudotojo.

Antrojo reikalavimo (ne)paisymas turi poveikį ta programavimo kalba parašytos taikomosios programos naudotojui netiesiogiai – per programavimo kalbos transliatorių, programuotoją ir jo parašytą taikomąją programą (56 pav.). Todėl ne visada lengva atsekti, kurioje grandyje yra programos neatitikimo kalbos ir kultūros terpei priežastis.

Jeigu programavimo kalba vartojama mokymui, tai tada programuotoju tampa mokinys (studentas), jis yra ir jo paties parašytos programos naudotojas. Tada lieka tik viena tarpinė grandis – transliatorius, – kurią mokymosi lygmenyje galima sutapatinti su programavimo kalba (žr. 56 pav. dešinįjį stulpelį).



56 pav. Ryšis tarp programų, programavimo kalbų ir jų naudotojų

Viena iš plačiausiai pasaulyje naudojamų programavimo kalbų, kurios bazinius žodžius įprasta lokalizuoti, yra Logo. Lietuvoje ji naudojama bendrojo lavinimo mokyklose mokant algoritmavimo žemesniųjų 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.

Su algoritmavimo ir programavimo pradmenimis supažindinami ne tik būsimi programuotojai. Šios žinios reikalingos daugelyje technikos krypčių ir apskritai struktūriniam, objektiniam, loginiam mąstymui ugdyti. Todėl mokymui naudojamų programavimo kalbų lokalizavimo poreikis panašus į mokymo arba bendrojo naudojimo programų lokalizavimo poreikį.

Lokalizuotų profesionalių programavimo kalbų nėra daug. Nedaug turime ir lietuviškos literatūros, reikalingos profesionaliam programuotojui. Todėl programuotojas priverstas skaityti ir suprasti anglų kalba. Dėl to programuotojai didesnio programavimo kalbų ir jų transliatorių lokalizavimo poreikio nejaučia. Netiesioginis (ne)lokalizuotų programavimo kalbų poveikis sunkiau pastebimas ir į jį ne visada atkreipiamas dėmesys.

Kai programa, parašyta programavimo kalba, lokalizuojama, kelyje iki lokalizuotos programos naudotojo atsiranda dar viena grandis – lokalizuotojas. Čia jis atlieka lokalizuojamos programos naudotojo vaidmenį ir jo darbui neigiamą įtaką turi internacionalizacijos klaidos lokalizuojamoje programoje, kurių priežastis yra neinternacionalizuota programavimo kalba.

Dar viena priežastis, kodėl vengiama lokalizuoti programavimo kalbas, yra galimos suderinamumo problemos tarp lokalizuotų kompiliatorių. Rimgaudas Laucius savo disertacijoje (2007) pateikia šios problemos sprendimą siūlydamas formalų lokalizuojamų elementų aprašo modelį. Pagrindinė idėja – įtraukti lokalizuojamus kompiliatorių suderinamumo elementus į lokalių aprašus. Taip pat siūloma standartizuoti bazinių ir integruotų vardų žodžių vertimus. Tai užkirstų kelią galimoms skirtingoms programavimo kalbų interpretacijoms jas lokalizuojant.

Į lokalių aprašus siūloma įtraukti šiuos leksikos elementus: bazinius žodžius, operacijų ženklus, skaičių formatą, skyrybos ženklus, integruotus vardus. Čia galima nurodyti panašią situaciją, kai klaviatūros klavišų pavadinimai pateikiami klaviatūros tvarkyklėje, iš kur juos gali pasiimti kiekviena programa (žr. 2 priedą).

Tam, kad visa tai būtų galima visiškai lokalizuoti, kompiliatoriuje turi būti naudojamas unikodas. Unikodo standartas apibrėžia ne tik universalų ženklų kodavimo

metodą, bet ir pateikia programavimo kalbose naudojamų vardų sudarymo sintaksės taisykles[1]. Jomis remiantis, vardai gali būti sudaromi ir iš lotynų abėcėlės, ir iš kitų raštų (kirilicos, graikų, arabų, kinų ir kt.) raidžių ar kitokių ženklų.

Programavimo kalbų lokalizavimas labai svarbus tautoms, kurios vartoja nelotynišką raštą ir kurių kalbos gerokai skiriasi nuo indoeuropiečių kalbų. Egzistuoja tamprus ryšys tarp žmogaus mąstymo ir kalbos. Žmogus vartoja kalbą, kad mintyse apibrėžtai išreikštų tai, ką galvoja. Panašiai elgiasi ir programuotojas: kurdamas programą mąsto programavimo kalbos, kuria programuoja, konstrukcijomis ir sąvokomis. Todėl labai svarbu, kad šios sąvokos ir jas žymintys leksikos elementai būtų kuo artimesni gimtajai kalbai (Laucius, 2007).

Programavimo kalbą apibrėžia jos gramatika – sintaksės ir semantikos taisyklių rinkinys. Tai norminis dokumentas, įforminamas oficialiu aprašu, dažniausiai vadinamu pranešimu apie kalbą. Jis laikomas pirminiu kalbos šaltiniu. Jei kalba paplinta, dažnai išleidžiamas jos nacionalinis arba tarptautinis standartas. Vadinasi, programavimo kalbos lokalizavimas yra jos pagrindinio dokumento vertimas ir adaptavimas konkrečiai kalbos ir kultūros terpei.

Programavimo kalbos sintakse nusakomos programos teksto formavimo taisyklės. Jos apibrėžiamos formaliai, dažniausiai Bekaus Nauro forma. Semantika skiriama programos veiksmams, užrašytiems sintaksės apibrėžtomis teksto konstrukcijomis, pateikti, dažniausiai tai daroma neformaliai.

Sintaksės taisyklėmis apibrėžiami terminaliniai ir neterminaliniai simboliai. Neterminaliniai simboliai reikalingi pačioms taisyklėms užrašyti, o formaliai žiūrint – programos tekstui generuoti. Programos tekstas sudaromas vien iš terminalinių simbolių. Programa pradedama generuoti nuo pagrindinio neterminalinio simbolio (dažniausiai įvardijamo žodžiu *programa*). Panaudojant sintaksės taisykles pagrindinis neterminalinis simbolis pakeičiamas kitų simbolių (terminalinių ir neterminalinių) seka. Joje esantys neterminaliniai simboliai ir toliau keičiami kitų simbolių sekomis, kol lieka vien terminaliniai simboliai: operacijų ženklai, skyrybos ženklai, baziniai kalbos žodžiai, o taip pat iš elementarių terminalinių simbolių (raidžių, skaitmenų ir kai kurių kitų ženklų) sudaryti vardai, skaičiai, ženklų eilutės. Visa tai matoma programos tekste ir sudaro kalbos leksiką. Todėl kalbos leksika yra pats svarbiausias dalykas, kurį reikia lokalizuoti.

[1] Unicode identifier and pattern syntax. Unicode Standard Annex #31. http://unicode.org/reports/tr31/

8.1.1. Leksikos lokalizavimas

Leksikos lokalizavimo galimybės buvo numatytos jau Algolo 68 kalboje. Jo oficialiame aprašyme (Wijngaarden, 1975) apie lokalizavimą arba internacionalizavimą nebuvo kalbama. Tačiau sintaksė buvo sudaryta taip, kad neterminalinių simbolių žymenis būtų galima parinkti kalbos realizacijoje transliatoriaus projektavimo metu.

Kiekvienas terminalinis simbolis turi pavadinimą. Sintaksės konstrukcijų apibrėžtys baigiamos terminalinių simbolių pavadinimais. Koks ženklas, ženklų grupė arba bazinis žodis atitinka tą simbolį, pateikta atskiroje aprašo dalyje. Laikoma, kad šios atitiktys priklausomos nuo realizacijos, t. y. jas galima pasirinkti realizacijos (transliatoriaus rengimo) metu. Taigi, galima parinkti terminalinių simbolių žymenis, atitinkančius kalbos ir kultūros terpę. Taip aiškiai atskiriama gramatikos taisyklių lokalizuojamoji dalis, panašiai kaip lokalizuojamieji ištekliai programoje. Todėl galima tvirtinti, kad ši kalba buvo internacionalizuota. Tuo pasinaudojus ji buvo lokalizuota į bulgarų, japonų, prancūzų, rusų ir vokiečių kalbas.

Operacijų ženklai. Matematikos žymenis, tarp jų ir operacijų ženklus, galima laikyti tarptautiniais, nes jie beveik tie patys visose kalbose, bent jau Europos šalyse. Todėl jų lokalizuoti nereikia, išskyrus vieną kitą specialų atvejį (pvz., ne visos tautos sveikųjų skaičių dalybą žymi ženklu ÷, loginių operacijų ženklai taip pat įvairūs). Suprantama, kad operacijų ženklai programavime turėtų būti tokie pat, kaip ir matematikoje. Tai būtų lyg ir sutartinis programavimo kalbose vartojamų operacijų ženklų internacionalizavimas (38 lentelė).

Turtingą operacijų ženklų rinkinį, perimtą iš matematikos, turėjo Algolo 60 kalba. Laikui bėgant situacija keitėsi: iš šešiolikos Algole 60 vartotų ženklų šiuo metu labiausiai paplitusiose programavimo kalbose C++ ir Java išliko tik penki: +, -, /, <, >. Iš jų – tik keturi tikri matematikos ženklai. Vietoj minuso vartojamas į jį panašus brūkšnelis. Kiti matematikos ženklai pakeisti ženklų poromis arba baziniais žodžiais. Kitokių negu matematikoje matematikos ženklų vartojimas programavime panašus į profesinį žargoną.

Tokią evoliuciją prastėjimo link lėmė matematikos ženklų mažinimas koduotėse. Algolo 60 kūrimo metu naudotose koduotėse jų buvo pakankamai (3 skyr.). Tačiau vėliau įsitvirtinusioje ASCII koduotėje, kuria buvo remiamasi kuriant naujas kalbas, jų nebeliko. Dabar, kai jau vartojamas unikodas, yra sąlygos grįžti prie matematikos žymenų ir ištaisyti nepagrįstus ženklų aibės ribojimus.

38 lentelė. Operacijų ženklai įvairiose programavimo kalbose

Operacija	Matematika (LT)	Kalba						
		Algol 60	Paskalis	Modula	Oberon	Ada 95	C++	Java
Sudėtis	+	+	+	+	+	+	+	+
Atimtis	–	-	-	-	-	-	-	-
Daugyba	×	×	*	*	*	*	*	*
Dalyba	/	/	/	/	/	/	/	/
Sveikųjų skaičių dalyba		÷	div	div	div	div	/	/
Sveikųjų skaičių dalybos liekana			mod	mod	mod	mod	%	%
Inversija	¬	¬	not	not ~	~	not	!	!
Konjunkcija	∧	∧	and	and &	&	and	&&	&&
Disjunkcija	∨	∨	or	or	or	or	||	||
Implikacija	⊃	⊃	<=					
Ekvivalentumas	≡	≡	=					
Neekvivalentumas	≢		<>			xor		
Mažiau	<	<	<	<	<	<	<	<
Ne daugiau	⩽	⩽	<=	<=	<=	<=	<=	<=
Lygu	=	=	=	=	=	=	==	==
Nelygu	≠	≠	<>	<> #	#	/=	!=	!=
Daugiau	>	>	>	>	>	>	>	>
Ne mažiau	⩾	⩾	>=	>=	>=	>=	>=	>=

Unikode yra nemažai kitų matematikos ženklų, kurie galėtų būti panaudoti programavimo kalbose, pavyzdžiui, ∈ (priklausomybė aibei, U+2208), √ (kvadratinė šaknis, U+221A).

Ženklų, kurių nėra klaviatūroje, rinkimą galima sutvarkyti labai paprastai: juos rinkti ženklų poromis, kad ir tomis pačiomis, kuriomis tie ženklai dabar pakeičiami. Programos tekstų rengyklė turėtų ženklų porą automatiškai pakeisti vienu tikru ženklu. Taigi rinkimas būtų toks pat, koks yra dabar, bet programos tekstą būtų maloniau skaityti. Beje, keleto ženklų porų keitimas veikia tekstų rengyklėje „MS Word“ (<= → ⩽, >= → ⩾), nors ši rengyklė nuo matematikos gerokai toliau negu programavimo kalbų tekstų rengyklės.

Unikode yra ir atskiras priskyrimo ženklas ≔ (jo kodas U+2190). Minėta tekstų rengyklė juo keičia ženklų porą : ir =. Tačiau šiam tikslui dar geriau tiktų vartoti rodyklę į kairę ← (U+2190).

Trupmenos skirtukas. Amerikiečiai ir anglai skaičiaus trupmeninę dalį nuo sveikosios skiria tašku, kitos Europos tautos – kableliu. Todėl internacionalizuotoje programavimo kalboje turi būti galimybė šį ženklą pasirinkti, lokalizacijoje reikia palikti tą ženklą, kuris vartojamas lokalizacijos terpėje.

Trupmenos skirtukas ypač svarbus duomenyse. Jeigu programavimo kalbos duomenų įvedimo procedūrose trupmenos skirtuko funkciją atlieka kitoks ženklas, negu vartojamas kompiuterio išorėje, t. y. kompiuterio naudojimo terpėje, tai nebus galima įvesti teisingai užrašytų realiųjų (trupmeninių) skaičių, o išvedimo procedūros spausdinimui pateiks neteisingai užrašytus realiuosius skaičius.

Programos operuoja su skaičiais, pavaizduotais vidiniais dvejetainiais kodais, kuriais skaičiaus semantika išreiškiama vidiniu kodavimu, nepriklausomu nuo matomo skaičiaus pavaizdavimo programose, jų duomenyse ar rezultatuose. Taigi skaičiavimai kompiuteryje nepriklauso nuo trupmenos skirtuko. Priklauso tik duomenų įvedimas (kol skaičiai perkoduojami iš išorinio pavaizdavimo ženklų eilutės į vidinį pavaizdavimą skaičiaus kodu) ir išvedimas (kol skaičiai perkoduojami iš vidinio pavaizdavimo į išorinį).

Naujesnėse programavimo kalbose (Deitsch, Czarnecki, 2001) duomenų įvedimas ir išvedimas išskiriamas į atskirus programos modulius, kurie gali būti keičiami. Tai analogiška lokalizuojamųjų išteklių atskyrimui. Taip daromas žingsnis programavimo kalbos internacionalizavimo link. Tačiau profesionaliomis programavimo kalbomis parašytuose programų tekstuose trupmenos skirtukas vis dar tebėra taškas. Jį mato tik programuotojas, o eilinis kompiuterio naudotojas programų neskaito. Todėl tiesioginio poveikio visuomenei jis nedaro. Tačiau netiesioginis poveikis per programuotoją aiškiai jaučiamas. Pavyzdžiui, Lietuvos programuotojas, nuolat matydamas tašką programose, ima įsivaizduoti, kad trupmeninė skaičiaus dalis iš tikrųjų skiriama tašku, ir ne tik pamiršta lokalizuoti trupmenos skirtuką, bet ir pats ima daryti klaidas rašydamas originalias programas. Kad taip iš tikrųjų yra, liudija dar dažnai pasitaikančios skaičių rašymo klaidos spaudoje, internete, netgi finansiniuose dokumentuose.

Programavimo kalbose kablelis atlieka ir sąrašo komponentų skirtuko funkciją (skiria, pvz., vardus aprašuose, parametrus kreipinyje į procedūrą). Todėl galima gauti nevienareikšmiškumų. Pavyzdžiui, užrašas 25,625 gali būti suprastas kaip kableliu atskirti du skaičiai 25 ir 625 arba kaip trupmena, sudaryta iš sveikojo skaičiaus 25 ir trupmeninės

dalies 625. Tačiau užrašas 25, 635 vienareikšmis, kadangi po kablelio yra tarpas, o tarpas skaičiaus viduje neleistinas. Taigi vienareikšmiškumas priklauso nuo tarpo. Geras programavimo stilius reikalauja, kad sąrašuose po kablelio (arba kitokio skirtuko, pvz., kabliataškio) būtų tarpas. Taigi laikantis gero programavimo stiliaus nevienareikšmiškumo bus savaime išvengta. Beje, programavimo kalbose ir taškas turi kitokių funkcijų: juo skiriami sudėtinių vardų komponentai, žymima programos pabaiga.

ISO standartai pripažįsta abu trupmenos skirtukus. Ir jų vartojimas apylygis. Beveik visos Europos ir Pietų Amerikos tautos vartoja kablelį, beveik visos Šiaurės Amerikos, Azijos ir Australijos tautos – tašką.

ISO 9995-7 standartas pateikia neutralų trupmenos skirtuko ženklą ⁄ (U+2396) skaitmeninės klaviatūros dalies klavišui žymėti.

Baziniai žodžiai. Beveik visų programavimo kalbų baziniai žodžiai paimti iš anglų kalbos arba vartojami jų trumpiniai. Buvo tikimasi, kad jie programavime atliks tarptautinę unifikavimo funkciją, analogišką lotyniškiems augalų pavadinimams botanikoje. Tada būtų galimybė vienareikšmiškai identifikuoti programų konstrukcijas. Deja, atsirandant naujoms kalboms baziniai žodžiai keitėsi. Aštuonių daugiau žinomų ir turėjusių įtakos programavimo raidai kalbų analizė (Grigas, 2000) parodė, kad tik septyni žodžiai (*do, else, false, for, if, true, while*) vartojami visose aštuoniose kalbose: Algole 60, Algole 68, Objektiniame Paskalyje, Moduloje, Komponentiniame Paskalyje, Adoje, C++ ir Javoje (39 lentelė).

Minėtus septynis bazinius žodžius galima laikyti universaliais – jie iš tiesų atlieka panašų vaidmenį, kaip ir lotyniški augalų pavadinimai botanikoje. Bet jų nedaug. Beveik pusė kiekvienos kalbos bazinių žodžių yra saviti, nepakartoti nei vienoje kitoje kalboje.

39 lentelė. Bendrų bazinių žodžių vartojimas keliose kalbose

Kalbų skaičius	Bendrų bazinių žodžių	
	skaičius	proc.
8	7	3,3
7	2	0,9
6	6	2,8
5	9	4,4
4	18	8,5
3	16	7,6
2	35	16,5
1	119	56,0
Iš viso	**212**	**100,0**

Anglų kalbos žodžiai turi mažai gramatinių formų. Tai buvo laikoma kalbos privalumu. Ankstesnėse programavimo kalbose (pvz., Algole 60, Paskalyje) baziniai žodžiai turėjo pagrindines anglų kalbos žodžių formas arba buvo jų trumpiniai. Vėliau šia anglų kalbos savybe buvo mažiau naudojamasi. Atsirado žodžių, turinčių galūnę *–s*, reiškiančią daugiskaitą, arba veiksmažodžio esamojo laiko trečiąjį asmenį, veiksmažodžių, turinčių galūnes *-ed*, *-ing* ir pan.

Anglų kalbos žodžių rašybos ir tarties sudėtingumas, taip pat tarties neatitikimas rašybai turi neigiamą poveikį bazinių žodžių aiškumui, ypač neanglakalbiam programavimo kalbos vartotojams.

Kai kuriose programavimo kalbose yra bazinių žodžių, vietoj kurių sinonimiškai galima vartoti specialiuosius ženklus arba jų poras. Pateiksime keletą pavyzdžių.

Moduloje vietoj bazinių žodžių *not* ir *and* galima vartoti jiems lygiaverčius sinonimus: ~ ir &. C++ kalboje yra net 10 sinonimų: *and (&&), bitor (|), or (||), xor (^) compl(~), bitand (&), and_eq (&=), or_eq (|=), not (!), not_eq (!=)*. Kartu žengtas dar vienas žingsnis lokalizavimo link – kalbos aprašyme šie baziniai žodžiai išvardyti atskirame sąraše – tai priduoda jiems žemesnio statuso atspalvį (38 lentelėje jie neparodyti).

Tam, kad žymenys būtų universalesni, kai kurie baziniai žodžiai keičiami ženklais:

begin → { (C, C++);
end → } (C, C++);
pointer → ^ (Paskalis).

Dalies bazinių žodžių pavyksta atsisakyti tinkamai parinkus kalbos sintaksės konstrukcijas. Pavyzdžiui, C, C++ ir Javos kalbos neturi žodžių *array*, *by*, *type*, *then*, nors konstrukcijas, kurios šiais žodžiais žymimos kitose kalbose, turi.

Kadangi matematikos ženklai mažai priklauso nuo natūralios kalbos, tai bet koks bazinių žodžių keitimas jais gerina programavimo kalbos internacionalizuotumo laipsnį. Tačiau tokių keitimų kol kas retai pasitaiko.

Bazinių žodžių būna kelios dešimtys (40 lentelė). Programuotojui, nuolat vartojančiam kalbą, juos nesunku įsiminti. Tačiau kai kalba vartojama epizodiškai, pavyzdžiui, algoritmavimo arba programavimo pradmenims mokyti, tai ir tie keliasdešimt žodžių – nemažas balastas. Todėl mokymui vartojamose kalbose jie verčiami. Juolab, kad ir vertimas nesudėtingas – nereikia derinti gramatinių formų, įvertinti konteksto. Nesudėtinga ir realizacija: tereikia į transliatorių įtraukti komponentą, įkeliantį bazinius žodžius iš lokalės (iš tikrųjų, jie ten turėtų būti).

40 lentelė. Bazinių žodžių skaičius programavimo kalbose

Kalba	Bazinių žodžių
Algolas 60	25
Algolas 68	58
Paskalis	43
Modula 3	62
Oberonas 2	45
Komponentinis Paskalis	53
Ada 95	75
C++	74
Java	59

Pastaba. Siekiant vienodumo į bazinių žodžių skaičių įtraukti ir 16-os duomenų tipų (*boolean*, *char* ir kt.), jų konstantų (*true*, *false* ir *null*) ir operacijos *abs* vardai, kai kuriose kalbose turintys integruotųjų vardų statusą.

Integruotieji (standartiniai) vardai. Tai, funkcijų, procedūrų, modulių, kai kuriose kalbose ir duomenų tipų, konstantų, įtrauktų į kalbą, vardai. Ankstesnėse kalbose jų buvo po kelias dešimtis, naujesnėse būna po kelis šimtus. Papildomai jų įtraukiama kalbų realizacijose (kompiliatoriuose). Jų statusas žemesnis, negu bazinių žodžių. Jie nebūna rezervuoti, todėl juos galima aprašyti iš naujo ir taip juos įvardyti naujais vardais. Tuo pasinaudojant į transliatorių galima įdiegti nesudėtingą mechanizmą, kuris tokius duomenų tipus ir konstantas aprašytų iš naujo, lokalizuotais vardais.

Šiek tiek sudėtingiau lokalizuoti integruotų funkcijų ir procedūrų vardus. Jos įvairiuose kompiliatoriuose realizuojamos įvairiai, tad sunkiau pasiūlyti universalų jų lokalizavimo mechanizmą. Tačiau daugumoje kompiliatorių integruotosios funkcijos žymimos vidiniais vardais, o tikrieji vardai atlieka tik leksikos elementų vaidmenį. Tokiais atvejais jie gali būti lengvai internacionalizuojami.

Ankstesnėse programavimo kalbose integruotaisiais vardais buvo žymimos matematinės funkcijos. Jų nebuvo daug, o vardai dažniausiai tarptautiniai (*sin*, *cos*, *lg* ir t. t.). Todėl nebuvo nė ką lokalizuoti.

Į naujesnes kalbas įtraukiamos ištisos funkcijų, procedūrų, modulių, klasių bibliotekos. Todėl ir integruotųjų vardų atsiranda daug, didėja jų vertimo poreikis. Apskritai, integruoti vardai verčiami dažniau, negu baziniai žodžiai.

Ne visada lengva nubrėžti ribą tarp bazinių žodžių ir integruotųjų vardų. Tas pats žodis vienoje programavimo kalboje gali būti laikomas baziniu žodžiu, kitoje – integruotuoju vardu.

Programose būna direktyvų vardų. Jie paprastai nėra aprašomi programavimo kalbų standartuose. Direktyvos skirtos valdyti kompiliavimo procesą, tad jos susijusios su konkrečiu kompiliatoriumi, todėl jų vardų pasirinkimo teisė priklauso kompiliatoriaus gamintojui ir paprastai direktyvos aprašomos kompiliatoriaus naudotojo dokumentacijoje.

Laisvai kuriami vardai. Tai konstantų, kintamųjų, duomenų tipų, funkcijų, procedūrų ir kiti vardai programoje sukurtiems objektams įvardyti. Esminė vardo savybė yra jo mnemonika – vardas turi atspindėti juo pažymėto objekto prasmę. Gerai parinkti vardai padeda suvokti, ką jie žymi, lengviau įsimenami.

Vardai sudaromi iš raidžių ir skaitmenų. Kai kurių programavimo kalbų varduose dar galima vartoti vieną kitą specialųjį ženklą, dažniausiai pabraukimo brūkšnį.

Raidžių aibė išvardijama arba kitaip nusakoma kalbos sintaksėje. Semantika neapibrėžia vardą sudarančių ženklų prasmės. Raidės tarnauja tik kaip „statybinė medžiaga“ vardams sudaryti. Vardai įgyja semantiką, bet ji nepriklauso nuo vardą sudarančių ženklų. Atskirai paimtos raidės semantika taip pat neapibrėžta. Todėl riboti raidžių aibę nėra pagrindo.

Pirmose programavimo kalbose buvo remiamasi ASCII koduote ir apsiribojama 26-iomis anglų kalbos abėcėlės raidėmis. Tokie ribojimai dar išliko ir atsiradus 8 bitų koduotėms.

Algolo 68 oficialiame angliškame aprašyme jau pasakyta, kad kalbos vertimuose raidžių aibė gali būti papildyta kalbinės terpės abėcėlės raidėmis. Moduloje 2 dar buvo tik anglų kalbos abėcėlės raidės, o Moduloje 3 – ir visi ženklai, esantys ISO 8859-1 kodų lentelės antroje pusėje. C kalboje tik angliška abėcėlė, o C++ – visos raidės ir hieroglifai, esantys unikode.

Tam, kad būtų galima turėti visavertę abėcėlę formaliai nenusižengiant galiojantiems kalbos standartams, į naujas programavimo kalbų standartų laidas įtraukiamas teiginys, jog konkrečioje realizacijoje (transliatoriuje) ženklų aibę galima papildyti ir bus laikoma, kad realizacija suderinta su standartu. Reikia tik tokius papildymus apibrėžti kalbos (transliatoriaus) dokumentacijoje.

Naujose programavimo kalbose remiamasi unikodu, ir tada jau nebereikia teiginio apie ženklų papildymą, nes nebėra kuo papildyti. Raidės su diakritiniais ženklais unikode gali būti užrašytos keliais būdais: raidės kodu (jeigu raidė jį turi), kompozicine seka (jeigu raidė turi vieną diakritinį ženklą) arba keliomis alternatyviomis sekomis (jeigu raidė turi kelis diakritinius ženklus) (3.4 skyr.). Skirtinguose to paties vardo pavartojimuose raidė gali būti užrašyta skirtingais būdais. Transliatoriui iškyla naujas uždavinys: atpažinti, kad vardas tas pats.

Mums tai nelabai aktualu, nes visos savitosios lietuvių kalbos abėcėlės raidės turi tik po vieną diakritinį ženklą ir unikodo kodus, nerenkamos kompozicinėmis sekomis. Nebent varduose imtume vartoti kirčiuotas raides, kas mažai tikėtina.

Leksikos elementų įkėlimas. Tam, 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 atskirame faile gali būti išsprendžiama kompiliatoriaus ir jam sukurtų programų perkeliamumo problema. Įkeliant skirtingų lokalių leksikos elementus dinamiškai, kompiliatoriaus veikimo metu, kompiliatorius taps nepriklausomas nuo konkrečios lokalės ir galės kompiliuoti įvairioms lokalėms skirtas programas (pavyzdžiui, parenkant atitinkamas direktyvas, kompiliatoriaus parinktis arba nurodant programos lokalę).

Suderinamumui su anksčiau sukurtomis programomis palaikyti tokiu atveju įmanoma įdiegti fiktyvią lokalę, į kurią būtų įtraukti leksikos elementai, atitinkantys nelokalizuoto kompiliatoriaus veikseną. Nustačius šią lokalę, būtų įmanoma sukompiliuoti anksčiau parašytas programas.

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 iškilti 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 pridėjus papildomų ženklų.

Kultūrinių skirtumų, įtakojančių programavimo kalbos leksiką, yra daugiau nei čia pateikta. Projektuojant leksikos lokalės modelį, nederėtų joje aprašomų elementų aibės laikyti fiksuota, tam, kad vėliau ją būtų galima lengvai išplėsti. Lokalėms aprašyti sėkmingai taikomas XML formatas. Jis gerai tiktų ir leksikos lokalei. Jo pranašumai: atvirumas, tinkamumas medžio tipo duomenų struktūroms aprašyti.

8.1.2. Semantikos lokalizavimas

Programavimo kalbų semantika nėra standartizuota, todėl ją sunku lokalizuoti. Kadangi semantikai priskiriami operacijų ženklai ir jie programavimo kalboje būna

vienareikšmiškai nusakyti, juos galima lengvai lokalizuoti. Visi kiti programavimo kalbų semantikos elementai reikalauja tolesnių išsamių mokslinių tyrimų.

Operacijos su ženklais. Operacijų su skaičiais semantika nepriklauso nuo kalbos ir kultūros aplinkos. Skaičius ir operacijas su jais visi supranta vienodai.

Rašto ženklų aibės (abėcėlės) skirtingose kalbose skirtingos, skiriasi ir raidžių rikiavimo eilė. Dėl to raidžių, bendru atveju ženklų, palyginimo rezultatai skirtingi. Pavyzdžiui, operacijos $Y < R$ rezultatas lietuvių kalboje yra loginė reikšmė *teisinga*, nes raidė Y eina prieš R, o lenkų kalboje – *neteisinga*, nes Y eina po R.

Oficialiuose programavimo kalbų aprašymuose iš šios situacijos išsisukama labai paprastai: pasakoma, kad tokios operacijos rezultatas neapibrėžtas. Realizacijoje (transliatoriuje) rezultatą ne tik galima, bet ir reikia apibrėžti, nes jeigu operacija yra, tai turi duoti kokį nors rezultatą. Apibrėžti galima dvejopai:

1. Laikantis konkrečios aplinkos kalbos taisyklių. Palyginimo operacijos semantikos aprašyme pateikiama konkrečioje aplinkos kalboje nustatyta rašto ženklų rikiavimo eilė.
2. Remiantis kompiuteryje veikiančia lokale. Palyginimo operacijos semantikos aprašyme pasakoma, kad ženklų rikiavimo eilė nustatoma pagal kompiuteryje veikiančioje lokalėje nustatytą raidžių rikiavimą.

Antrasis atvejis universalesnis. Jį galima laikyti programavimo kalbos internacionalizavimo elementu.

Programavimo kalba ir kompiliatorius. Programavimo kalbos naudojimas neatsiejamas nuo kompiliatoriaus. Programuotojas praktiškai gali pasinaudoti tik tomis programavimo kalbos galimybėmis, kurios realizuotos kompiliatoriuje. Jų gali būti ir daugiau, negu buvo pateikta programavimo kalboje. Bet gali likti ir nerealizuotų. Todėl faktiškai programuotojas dirba su tuo kalbos variantu (dialektu), kuris realizuotas kompiliatoriuje. Oficialus kalbos aprašymas dažniausiai būna tik kelrodis kitų kompiliatorių kūrėjams, kad jie turėtų orientyrą ir stengtųsi kuo mažiau nuo jo nukrypti.

Tai, kas buvo neapibrėžta programavimo kalboje, apibrėžiama kompiliatoriuje – jeigu atliekama operacija, tai rezultatas turi būti, nors ir neteisingas. Su ženklų operacijomis dažnai taip ir atsitinka. Paprasčiausia palyginti ženklų kodus. Kai kurių kompiliatorių autoriai tuo ir pasinaudoja. Juolab, kad anglų kalbos abėcėlės raidžių kodai ASCII ir jos pagrindu sudarytose aštuonių bitų koduotėse išdėstyti iš eilės. Todėl angliški tekstai rikiuojami teisingai, o kitų kalbų – neteisingai. Tai šiurkšti kompiliatoriaus internacionalizavimo klaida. Iš tikrųjų klaidos šaknys giliau – koduotėse. Tai, kad kodų eilė pritaikyta tik vienos kalbos, ir dar plačiai vartojamos, abėcėlei, sudaro sąlygas kompiliatorių kūrėjams nepasirūpinti kitomis kalbomis.

Ženklų tipai. Ženklų tipo reikšmė – bet koks ženklas iš kompiuteryje naudojamos koduotės. Iki unikodo ženklui buvo skiriamas vienas baitas. Situacija pasikeitė atsiradus unikodui: vienam ženklui prireikė dviejų baitų. Bet daug, gal net dauguma, tekstų rašomi 8 bitų koduotėmis ir jų ženklams skiriama po vieną baitą. Todėl dabar reikia dviejų ženklų tipų: vienbaičių ir dvibaičių (analogiškai kaip skaičiai – įprasti ir dvigubai ilgesni).

Unikodo ženklai dažnai koduojami UTF-8 kodu – vienas unikodo ženklas užrašomas 1, 2, 3 arba 4 aštuonbitės koduotės ženklais (kodais). Tada unikodo tekstas virsta aštuonbičiu tekstu. Jeigu tekste vien anglų kalbos abėcėlės raidės, tai jis sutampa su aštuonbičiu tekstu, o tuomet yra pavojus supainioti koduotes ir ne taip interpretuoti tekstą.

8.1.3. Lokalizuotos kalbos

Lokalizuotų programavimo kalbų nėra daug. Lietuvių kalbai lokalizuotos trys Logo versijos: „Logo Writer“, Komenskio Logo (Comenius Logo) ir „Imagine Logo“. Tai populiariausia mokymui skirta kalba, lokalizuojama į daugelį kalbų. Iš dalies sulietuvinta Paskalio šeimos kalba „Free Pascal“: varduose galima vartoti visas lietuvių kalbos abėcėlės raides, į lietuvių kalbą išversti transliatoriaus pranešimai (Laucius, 2005). Tačiau nelokalizuotos operacijos su tekstu, neišversti baziniai žodžiai. Mokymui skirtos Paskalio kalbos versijoje Algo[1] galima atskirai pasirinkti bazinių žodžių ir sąsajos kalbą.

Vienas iš žinomiausių pavyzdžių – kinų kalbai lokalizuota programavimo kalba „Python“ (中蟒 (中文 Python)[2]). Išversti ne tik baziniai žodžiai, bet ir pakeisti daugelio operacijų ženklai vartojamais Kinijoje.

Kartu su programavimo kalba reikia lokalizuoti ir jos transliatorių. Internacionalizuotų transliatorių dar nedaug. Todėl lokalizuojant kalbą dažniausiai tenka parašyti ir naują transliatorių. Jį berašant atsiranda galimybė patobulinti kalbą, nukrypstama nuo originalios kalbos. Tada sakoma, kad sukurta nauja kalba tos originalios kalbos pagrindu. Taip gimsta nauja programavimo kalba. Pavyzdžiui, „Phoenix“[3] (arabų, C kalbos pagrindu), „Glagol“[4] (rusų, Oberono pagrindu), „Jeem“[5] (arabų, C++ pagrindu).

---

[1] Portal edukacyjny – Informatyka. http://www.informatyka.zsp-slawa.pl/index.php?option=com_docman&task=cat_view&gid=1&Itemid=13

[2] Chinese Phyton, http://www.chinesepython.org/cgi_bin/cgb.cgi/home.html

[3] Phoenix Arabic Programming Language, http://sourceforge.net/projects/phoenix/

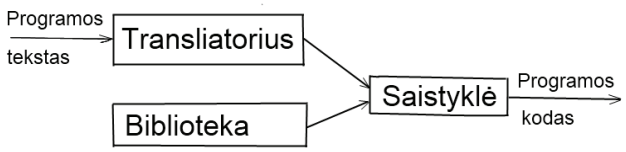
[4] Разработки на Глаголе, http://glagol.nad.ru/

[5] S. Sarham. Arabic Programming Languages, University of Jordan, http://ipac.kacst.edu.sa/eDoc/2007/163506_1.pdf

8.2. Kompiliatoriai

Transliatorius atlieka programavimo kalba (dažniausiai suprantama žmogui) užrašytos programos vertimą į kompiuteriui suprantamą kito lygio kalbą (kompiuterio kalbą, kuria pateiktas programos tekstas, neretai vadinamas mašininiu kodu). Profesionaliuose tekstuose vartojama ne tik kompiliatoriaus, bet ir transliatoriaus sąvoka. Pagrindinė savybė, skirianti kompiliatorių nuo transliatoriaus, yra ta, kad kompiliatorius į vertimą įkomponuoja jau parengtų programos fragmentų (bibliotekų). Transliatorius yra siauresnis terminas, įvardijantis tik programos vertimą. Transliatoriaus ir kompiliatoriaus atliekamos funkcijos vis labiau panašėja, nes nebeįsivaizuojama transliatoriaus be bibliotekų ir papildomų modulių, todėl praktikoje transliatoriaus ir kompiliatoriaus terminai vartojami sinonimiškai. Transliatoriaus terminas vartojamas siaurąja prasme įvardijant tą kompiliatoriaus dalį, kuri skirta programai išversti į kompiuterio naudojamą žemesnio lygio kalbą.

Pirmiausia trumpai aptarsime kompiliatoriaus dalis, jų ryšius, veikimo principus. Kompiliatorių paprastai sudaro trys pagrindinės dalys: 1) transliatorius, 2) vykdymo meto biblioteka ir 3) saistyklė (57 pav.).



57 pav. Šiuolaikinio kompiliatoriaus sandara ir veikimo schema

Saistyklė, naudodama transliatoriaus sugeneruotuose failuose esančią informaciją ir iš anksto parengtus programų fragmentus, laikomus bibliotekose, sugeneruoja galutinius kompiliatoriaus išvedinius (vykdomuosius failus arba bibliotekas). Dažniausiai jie išreiškiami kompiuteriui suprantamos programavimo kalbos kodais (mašininiais kodais), tarpinės kalbos kodais, suprantamais kitiems kompiliatoriams arba virtualiosioms mašinoms. Pavyzdžiui, „Visual C++“ ir „Delphi“ kompiliatoriuose transliatoriai ir saistyklės yra atskiros, savarankiškos programos. Kituose kompiliatoriuose jie gali būti tarpusavyje integruoti į vieną programą, pavyzdžiui, Java kompiliatoriaus saistyklė nėra aiškiai išreikšta – ja laikytinos Java platformos priemonės, skirtos Java baitinėms programoms tikrinti ir klasėms įkelti. Java kompiliatoriaus

atveju, skirtingai nuo „Visual C++“ ir „Delphi“ kompiliatorių, kurie atlieka susaistymą kompiliavimo metu, susaistymas atliekamas programos vykdymo metu.

Kompiliatoriaus ir programuotojo darbą paspartina iš anksto parengtų programų fragmentų, laikomų bibliotekose, panaudojimas. Transliatoriui jų nebereikia versti, o programuotojui nebereikia jų programuoti. Todėl šiuolaikiniai kompiliatoriai komplektuojami su įvairios paskirties programų fragmentų bibliotekomis. Svarbiausia iš šių bibliotekų yra vykdymo meto biblioteka. Ji realizuoja semantinę programavimo kalbos konstrukcijų dalį, integruotąsias procedūras, funkcijas ir kitus kompiliatoriaus veikimui būtinus elementus. Vykdymo meto biblioteka yra kompiliatoriaus dalis, be kurios jis negalėtų veikti.

Kompiliatorių lokalizavimas glaudžiai susijęs su internacionalizavimu: pramoninius kompiliatorius kuria tarptautinės firmos ir paprastai jie rašomi anglų kalba. Toliau šalys lokalizuoja pagal poreikius.

Transliatorius, vykdymo meto biblioteka ir saistyklė sudaro nuoseklią grandinę: transliatorius realizuoja sintaksinę kompiliatoriaus dalį, biblioteka – semantinę, o saistyklė – galutinį kodo generavimą. Todėl kompiliatoriaus lokalizavimui svarbu, kad visos šios trys dalys būtų pakankamai internacionalizuotos. Tačiau svarbiausią vaidmenį kompiliatorių internacionalizacijoje vaidina vykdymo meto bibliotekos internacionalizacija. Joje esantys programos teksto fragmentai kompiliavimo metu perkeliami į programą, todėl nuo to, kiek jie internacionalizuoti, priklauso tai, kiek internacionalizuota bus kompiliatoriumi sukurta programa.

Kompiliatorių lokalizavimo procesą sudaro kelios pakopos:

1. Kompiliatoriuje naudojamos informacijos, tiksliau – duomenų – lokalizavimas.
2. Bazinių žodžių lokalizavimas.
3. Kompiliatoriaus sąsajos lokalizavimas.
4. Kompiliatoriaus žinyno vertimas.

Pirmoji dalis specifiškiausia, nėra bendrų sutarimų, kuri informacija reikalinga vienam ar kitam kompiliatoriui. Bazinių žodžių lokalizavimą nusako programavimo kalba, kurią realizuoja kompiliatorius. Sąsajos lokalizavimas iš esmės nesiskiria nuo bet kurių kitų programų lokalizavimo, tas pats sakytina ir apie žinynų vertimą.

Kompiliatoriaus duomenų lokalizavimas. Kompiliatorius išoriniams duomenims (pirminiam tekstui, parinktims ir kitiems failams) koduoti turėtų naudoti unikodą. Svarbu, kad duomenys nebūtų iškraipomi ar sugadinami juos apdorojant, pavyzdžiui, dėl klaidingai atliekamo perkodavimo ar kitų veiksmų, susijusių su programinės įrangos internacionalizavimu. Duomenys taip pat turi būti teisingai išvedami.

Jei kompiliatorius tinkamai suprogramuotas, t. y. jis internacionalizuotas, tuomet nesunku jį lokalizuoti, nes kompiliatoriaus naudojami tekstiniai ištekliai atskirti nuo veiksmų dalies. Tuomet nesunku derinti žodžių gramatines formas, į verčiamą tekstą įterpti ženklų kodus ir pan.

Vienodoms išteklių eilutėms turi būti priskirti skirtingi identifikatoriai. Eilutėms įterpti ir jungti naudojamos formatavimo funkcijos. Verčiant eilutes, pageidautina išversti ir komentarus, paaiškinančius jų vartojimo kontekstą. Jei komentarų nėra, reikėtų jais papildyti lokalizuojamas sudėtingesnes eilutes.

Tas pats tinka ir kompiliatoriaus naudojamiems netekstiniams ištekliams – jie turi būti atskirti nuo veiksmų dalies. Netekstiniais ištekliais laikomi sparčiųjų klavišų deriniai, paveikslai, piktogramos, žymekliai, garso ir vaizdo failai, dialogų šablonai (langų išdėstymas, spalvos ir kiti 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.

Jei kompiliatorius pakankamai internacionalizuotas, tai vykdymo meto bibliotekoje naudojami pranešimai kompiliavimo metu bus atskiriami nuo veiksmų dalies – tada juos bus galima lokalizuoti po to, kai jie bus įkompiliuoti į išvedinį.

Kompiliatoriaus realizuota įvedimo paslauga turi leisti įvesti unikodu koduotą tekstą – jei to nėra, kai kurioms kalboms gali būti labai sunku lokalizuoti kompiliatorių, reikės papildomų programavimo pastangų, teks numatyti perkodavimo galimybę, jei duomenys būtų įvedami iš unikodo nepalaikančio srauto. Perkodavimas turi būti atliekamas pagal nustatytoje lokalėje galiojančias nuostatas. Taip pat turi būti numatyta galimybė įvesti ženklus surenkant jų kodus paspaudus alternatyvos klavišą.

Analogiškai 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 pagal nustatytoje lokalėje galiojančias nuostatas. Lokalizuojant kompiliatorių reikia atkreipti dėmesį į tekstų vaizdavimo galimybes. Viskas gerai, jei kompiliatoriaus kūrėjai numatė ir įprastų, ir kompleksinių, ir dvikrypčių bei CJK raštų vaizdavimą, jei teisingai vaizduoja kombinacinių ženklų sekas. Jei tai nenumatyta, tada teks įdėti daugiau darbo priklausomai nuo lokalizavimo kalbos.

Jei kompiliatoriuje numatyta išplėtimo galimybė vykdymo meto bibliotekos ir sukompiliuotų programų lygiu, tai bus nesunku atlikti ir sudėtingesnius lokalizavimo veiksmus, pavyzdžiui, lokalizuojamą programą papildant nestandartinių šriftų (naudojančių kirčiuotas raides ar pan.) vaizdavimo galimybėmis.

Bazinių žodžių lokalizavimas. Bazinių žodžių vertimo problemas išnagrinėjome rašydami apie programavimo kalbas (8.1 skyr.). Tinkamai sukurtuose kompiliatoriuose baziniai žodžiai, direktyvų vardai, modifikatoriai, operacijų ženklai turi būti atskirti nuo veiksmų dalies – tada juos bus galima lengvai lokalizuoti. Jei tai nepadaryta, lokalizuojant tenka skaityti kompiliatoriaus programą ir ieškoti, kur paslėpti baziniai žodžiai.

Kompiliatoriai skirtingai realizuoja vardų sintaksę. Nors 1999 metų atnaujintas C++ kalbos standartas jau apibrėžia unikodo ženklų naudojimą varduose, tačiau, pavyzdžiui, *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, be to, jis nurodytas kompiliatoriaus dokumentacijoje. *Delphi .NET* kompiliatorius taip pat leidžia varduose naudoti unikodo ženklus, tačiau jų sintaksės nesuderina su Paskalio kalbos taisyklėmis, nes nėra atsižvelgiama į ženklų savybes (pavyzdžiui, vardai gali prasidėti lietuviškomis kabutėmis arba skaitmenimis).

Kompiliatoriaus sąsajos lokalizavimas. Dauguma šiuolaikinių kompiliatorių naudoja grafines sąsajas ir netgi vieną sąsają gali naudoti keli kompiliatoriai. Iš tikrųjų kompiliatoriaus sąsaja niekuo ypatingu nepasižymi, tiesiog ją galima lokalizuoti kaip bet kurios programos sąsają (apie tai išsamiai rašoma 6.2.1 skyr.).

Kompiliatoriaus sąsaja – tai meniu punktų ir komandų pavadinimai, užrašai ant mygtukų, mygtukų etiketės, kompiuterio pranešimai ir visi kiti tekstai, matomi kompiuterio ekrane veikiant kompiliatoriui. Dauguma užrašų – trumpi, lakoniški, sudaryti iš vieno arba kelių žodžių. Pranešimai paprastai būna trumpi sakiniai.

Sąsajos tekstai paprastai išdėstomi dialogo languose – būdinga tai, kad veikiant programai užrašų vieta lange gali keistis, tas pats tekstas gali atsidurti ir kitame lange, ir greta kitų tekstų. Sąsajos tekstai ištekliuose pateikiami atskiromis eilutėmis, todėl nematomas kontekstas, ir lokalizuotojui sunku teisingai išversti tokius tekstus. Neretai greta išteklių eilučių būna pateikti komentarai – paaiškinama, kokiame kontekste bus matomas tekstas. Tačiau kompiliatoriaus kontekstas specifinis, lokalizuotojas, norėdamas gerai išversti, turi neblogai išmanyti programavimą ir kompiliatorių veikimo principus.

Kompiliatoriaus žinyno vertimas. Kompiliatoriaus žinynas – tai visų pirma programavimo kalbos konstrukcijų aprašas, paprastai su išsamiais pavyzdžiais ir nuorodomis, ir tik paskui aprašomos kompiliatoriaus funkcijos, savybės, parametrai, nurodoma, kaip derinti, testuoti programas, kokias kompiliatoriaus direktyvas pasirinkti ir pan. Kompiliatoriaus žinyno vertimas beveik niekuo nesiskiria nuo bet kurios kitos programos žinynų vertimo, nebent tik tiek, kad pats kompiliatorius yra specifiškesnė programa, tad ir jo žinynas iš vertėjo reikalauja daugiau technikos ir informatikos žinių. Išsamiau apie žinynų lokalizavimą rašoma 6.2.1 skyriuje.

Kompiliatorių lokalizavimo tyrimas. R. Laucius disertaciniame darbe (2007) atliko trijų kompiliatorių tyrimą: 1) Visual C++ (Microsoft Visual Studio 2003 .NET)[1], 2) Java (Sun Java 1.5.0)[2] ir 3) Delphi (Borland Delphi, 2005). Buvo nustatyta, kad nei viename iš šių kompiliatorių nėra realizuota patogių bazinių žodžių, direktyvų vardų, modifikatorių, operacijų ženklų, skaičių ir skyrybos lokalizavimo galimybių, todėl kompiliatorių lokalizuotojai turi sugaišti daug laiko įvairiai spręsdami šias problemas.

Apibendrinus kompiliatorių tyrimą nustatyta, kad:

- Visų nagrinėtų kompiliatorių internacionalizuotumo lygis gana žemas.
- Visi kompiliatoriai pirminio teksto kodavimui leidžia naudoti unikodą, bet jo palaikymas yra ribotas.
- Visi kompiliatoriai turi trūkumų, susijusių su teksto įvedimo ir išvedimo realizacija. Pagrindiniai šių operacijų trūkumai susiję su naudojamu teksto kodavimu 8 bitais.
- Visi kompiliatoriai naudoja skirtingus išteklių atskyrimo metodus. Visi šie metodai turi ir pranašumų, ir trūkumų.
- Visų kompiliatorių realizuotos teksto apdorojimo funkcijos nesuderintos arba nepakankamai suderintos (tik *Java* kompiliatoriaus) su unikodo standartu.
- Nei vienas iš kompiliatorių, išskyrus *Java*, nerealizuoja arba realizuoja nepakankamai gerai kultūrinių nuostatų įkėlimo ir su jomis susijusių funkcijų.
- Tik *Java* kompiliatorius leidžia pirminiame tekste naudoti vardus, suderinamus su unikodo standartu ir atitinkančius programavimo kalbos taisykles.
- Nei vienas kompiliatorius neleidžia lokalizuoti programavimo kalbos leksikos.
- Visi kompiliatoriai naudoja tik 8 bitais koduotus failų vardus ir neteisingai arba nepakankamai gerai realizuoja failų sistemos paslaugas.

[1] Skliausteliuose pateikiamas PĮ paketo pavadinimas ir versija, kuriam priklauso kompiliatorius.

[2] „Java“ yra Sun Microsystems bendrovės registruotas prekės ženklas.

Kompiliatorius, kaip ir bendrosios paskirties programinė įranga, 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 programinės įrangos internacionalizuotumui (pavyzdžiui, unikodo duomenų tipų realizacija).

Internacionalizuojant kompiliatorius siekiama, kad jais kuriama programinė įranga būtų internacionalizuota, todėl svarbu:

- Vykdymo meto bibliotekoje įdiegti programinės įrangos internacionalizavimui skirtas paslaugas.
- Išplėsti vykdymo meto biblioteką tam, kad joje esantys kodo fragmentai būtų suderinami su įvairiomis lokalėmis.
- Vykdymo meto bibliotekoje įdiegti išteklių atskyrimo funkciją.
- Išplėsti kompiliatorių tam, kad jis būtų suderinamas su įvairiomis lokalėmis.

Kompiliatoriaus, kaip ir bendrosios paskirties programinės įrangos, lokalizuotini ištekliai turi būti atskirti nuo veiksmų dalies. Ypatingai svarbu, kad nuo veiksmų dalies būtų atskirti vykdymo meto bibliotekoje naudojami lokalizuotini ištekliai, nes nuo to priklauso ir tai, ar šie ištekliai bus atskirti kompiliatoriumi sukurtose programose.

8.3. Svetainės

Palyginus su bendrosios programinės įrangos lokalizavimu, interneto svetainių lokalizavimas yra ypatingas tuo, kad:

1. Susiduriama ne tik su eilučių (frazių) lokalizavimu ir lokalės elementų adaptavimu, bet ir su didelių tekstų kiekių vertimu (Maroto, Bortuli, 2001). Saityno tekstai ir jų lingvistinė specifika yra įvairių tyrimų objektas (pvz., Crystal, 2001).
2. Adaptuojamas žymiai didesnis kiekis grafikos, garsų, vaizdų, spalvų ir kitų mažiau akivaizdžių, nuo kultūros priklausomų, elementų (Maroto, Bortuli, 2001).

3. Dalis svetainės turinio ir struktūros projektuojama iš naujo taip, kad tiktų tam tikros kultūros svetainės lankytojams (Esselink, 2000).
4. Svetainės yra žymiai dažniau atnaujinamos, lyginant su kliento kompiuteriuose veikiančiomis programomis, taigi ir svetainės lokalizaciją būtina nuolat operatyviai atnaujinti.
5. Svetainių turinys dažnai formuojamas dinamiškai, o jų lokalizavimas yra susijęs su pagalbinės serveryje veikiančios programinės įrangos lokalizavimu.

Kadangi programinės įrangos dokumentacija, elektroniniai žinynai, žinių bazės ir kiti papildomi ištekliai vis dažniau pateikiami internete svetainių pavidalu, tai jų lokalizavimo procesas yra analogiškas svetainių lokalizavimo procesui.

Lokalizuojant svetaines nuolat susiduriama su kultūros elementų, aprašytų 4.2 skyrelyje, adaptavimu. Be abejo, svetainių lokalizavimas priklauso nuo ženklinimo kalbos (pvz., HTML, XML), scenarijų ar programavimo kalbos (pvz., PHP, „JavaScript“, „Phyton“) ir svetainių kūrimo technologijų, kuriomis remiantis realizuota svetainė, panašiai, kaip klientų programinės įrangos lokalizavimas priklauso nuo lokalizuojamųjų išteklių pateikimo formato (5.2 skyr.).

Toliau analizuosime pagrindinių svetainės elementų adaptavimo aspektus, o tada išskirsime svetainės turinio ir pagalbinės programinės įrangos lokalizavimo klausimus.

Antraštinė tinklalapio dalis. Šioje hipertekstinio dokumento dalyje rašoma tinklalapio metainformacija, kurios tiesiogiai nesimato tinklalapio tekste: tinklalapio pavadinimas, kuris bus matomas naršyklės lango antraštės juostoje, koduotė (pvz., Windows-1257, UTF-8), tinklalapio autoriaus asmenvardis arba organizacijos pavadinimas, tinklalapio reikšminiai žodžiai ir turinio anotacija (šiais duomenimis pasinaudoja paieškos sistemos), nuorodos į tinklalapyje naudojamų scenarijų, pakopinių stilių failus ir kita informacija.

Tinklalapio teksto koduotė nurodoma antraštės dalyje CHARSET požymiu:

```
<META HTTP-EQUIV="Content-Type"
 CONTENT="text/html; CHARSET=Windows-1257">
```

Pasirenkama tinklalapio kalbai tinkama koduotė: jei tinklalapyje naudojami įvairių kalbų rašmenys, svetainė yra daugiakalbė, tai dažniausiai bus UTF-8 koduotė, jei tinklalapyje vartojami tik vienos kalbos rašmenys arba kelių kaimyninių lokalių, naudojančių tą pačią 8 bitų koduotę, rašmenys, tai gali būti naudojama 8 bitų koduotė (pvz., Baltijos šalims – „Windows-1257“, ISO 8859-13).

Pagrindinė tinklalapio dalis. Tai tinklalapio tekstas, į jį įterptos lentelės, saitai, gairės ir kiti elementai. Čia taip pat gali būti įvairių formatavimo elementų: spalvų, šriftų, lygiavimo, kuriuos lokalizuojant gali tekti pakeisti. Dažnai lokalizuojama pasitelkus lokalizavimo dalinio automatizavimo priemones (7 skyr.).

Komentarai. Tai tekstas, tiesiogiai nematomas dokumente, tačiau naudingas tinklalapio kūrėjams arba lokalizuotojams, pavyzdžiui, gali būti įterpto scenarijaus ar dialogo eilutės paaiškinimas. Reikėtų palikti komentarus originalo kalba ir lokalizuojant juos papildyti naujais komentarais, susijusiais su vertimu į konkrečią kalbą, kad būsimi lokalizuotojai galėtų jais pasinaudoti.

Susietieji arba įterptieji scenarijai. Sudėtingesni veiksmai svetainėse dažnai realizuojami scenarijais: „JavaScript“, PHP, „VBScript“ ir kt. Juose dažniausiai būna lokalizuojamųjų išteklių – sąsajos teksto eilučių.

Grafika ir kiti įvairialypės terpės elementai. Paveikslai ir kiti grafikos elementai (pvz., piktogramos) ne visada tinka visoms lokalėms (ypač, jei paveiksle yra tekstas arba turi tam tikrą religinį, politinį atspalvį). Todėl, jei reikia, turi būti lokalizuojami.

Problemų lokalizavimo metu kelia svetainės simboliai ir piktogramos, skirtingai suprantamos įvairiose kultūrose (Choong, Salvendy, 1998; Evers, 2001b). Dauguma paveikslų ar piktogramų turi savyje subtilių kultūrinių pranešimų, pavyzdžiui, tam tikros organizacijos svetainėje nuotrauka vadovo, pavaizduoto vieno, be darbuotojų, netiks šalims, turinčioms aukštą kolektyvizmo dimensijos laipsnį, o turizmo agentūros svetainei, lokalizuojamai šalims, kurių pagrindinė religija yra islamas, netiktų žmonių su maudymosi kostiumais nuotraukos.

Piktograma, skirta pažymėti svetainės pradžios tinklalapį (anglų kalbos tinklalapiuose dažniausiai namo piktograma, nes angliškai pradžios tinklalapis yra „Home“) arba elektroninio pašto paslaugą (dažniausiai – vokas) tinka daugumai Vakarų šalių, bet netiks kai kurioms Rytų šalims. Lokalizuojant tokius elementus vadovaujamasi lokalės tradicijų žiniomis ir etikos taisyklėmis (9 skyr.).

Lokalizuojamas taip pat grafikos alternatyvusis tekstas ir etiketės, atsirandančios užvedus pelės žymeklį. Jei svetainėje naudojami tekstų įgarsinimai, tai paruošiami nauji įgarsinimai lokalės kalba. Jei svetainėje naudojami kiti garso failai, jie gali būti adaptuojami (pakeičiami) atsižvelgiant į kultūrines tradicijas.

Stiliai. Lokalizuota svetainė lankytojui turi atrodyti natūraliai, lyg jo kultūrinėje terpėje sukurtas produktas. Turi reikšmės ir lokalei priimtinos spalvos, šriftai, teksto išdėstymas ir kiti stilių elementai.

Dialogo laukų eilė. Interneto svetainėse paprastai naudojami atskiri laukai (langeliai) ar valdikliai duomenims, kurie priklauso nuo lokalės, įvesti. Pavyzdžiui, adreso įvedimo tinklalapyje gali būti daugeliui lokalių nereikalingas laukas „Valstija“, datai ir laikui nurodyti pateikiami trys atskiri langeliai: diena, mėnuo, metai (lokalizuojant reikia pakeisti datos langelių tvarką į atvirkščią – tam teks keisti tinklalapio ar scenarijaus pirminių tekstų fragmentą).

Jeigu svetainėje naudojama lankytojų registracijos ir prisijungimo funkcija, tai turi būti užtikrinta lokalės rašmenų vartojimo galimybė abonento varde, slaptažodyje, asmenvardyje. Jeigu iš naudotojo vardo ir pavardės automatiškai generuojami asmenvardžiai, tai turi būti naudojama teisinga vardo ir pavardės eilė (pvz., Vardas Pavardė).

Jei svetainė leidžia lankytojams siųsti failus, tai turi būti užtikrintas nacionalinių rašmenų vartojimas failų varduose ir jų teisingas apdorojimas. Jeigu iš svetainės galima parsisiųsti failus, tai jų vardai turėtų būti lokalizuojami.

Automatiškai įterpiamų datos, laiko, skaičių (pvz., failų dydžių), pinigų sumų, laiko juostos formatai turėtų būti parenkami vadovaujantis lokalės normomis.

Svarbus yra ir interneto srities vardas, kuris parenkamas lokalizuotai svetainei. Jei jame yra raidžių ne iš pagrindinės lotynų abėcėlės, tai reikėtų registruoti internacionalizuotą srities vardą (4.2 skyr.). Tinklalapio adresuose visi vartojami failų ir aplankų vardai turi būti išversti į svetainės kalbą ir parašyti vartojant visus tos kalbos rašmenis. Raidės, kurių nėra ASCII koduotėje, užrašomos UTF-8 kodu ir koduojamos pakaitos sekomis %FF (3.5.1 skyr.).

Projektuojant ir lokalizuojant sudėtingą daugiakalbę svetainę (pvz., elektroninės parduotuvės ar elektroninės bankininkystės sistemą) turi būti numatyti ne tik minėti kultūros elementai, bet ir atitinkamos vietinės paslaugos (pvz., prekių platintojai, pristatymas). Atsižvelgiant į rinkos ir lankytojų poreikius, kartais reikia visiškai pakeisti svetainės turinį, t. y. parašyti naują tekstą. Skiriasi ne tik kalba, bet ir kultūra, tradicijos, rinka, kuri daro įtaką svetainės turiniui, struktūrai ir pan. Dažnai kultūriniai dalykai yra ignoruojami, tokiu atveju susiduriama su prastai lokalizuotomis svetainėmis, kurios kenkia ne tik įmonės prestižui, bet ir verslo sėkmei (Adomavičiūtė, 2009).

Daugumos dabartinių svetainių lokalizavimas yra kelių lygių, kurie išplaukia iš lokalizuojamos svetainės struktūros:

- Turinio valdymo sistemos lokalizavimas.
- Svetainės turinio lokalizavimas.

Turinio valdymo sistemos lokalizavimas. Šios sistemos naudojamos svetainėms kurti, jos padeda sparčiai realizuoti dažnai naudojamus techninius svetainių aspektus: lankytojų registraciją, prisijungimą, svetainės turinio įdėjimą, modifikavimą ir šalinimą, turinio versijų sekimą ir kt. Turinio valdymo sistemos lokalizavimas yra labiau panašus į bendros paskirties programinės įrangos lokalizavimą: lokalizuojamieji ištekliai atskiriami nuo programos tekstų ir pateikiami lokalizuotojams vienu iš lokalizuojamųjų išteklių formatų (5.2 skyr.).

Svetainės turinio lokalizavimas. Svetainės turinys gali būti pagrindinis, paprastai įdėtas svetainės kūrėjų (pvz., informacija apie tam tikrą organizaciją, jos prekių katalogai, mokomoji medžiaga), ir papildomas, sukurtas svetainę lankančių žmonių ar jų grupių (pvz., lankytojų parašyti svetainės straipsnių komentarai, forumų pranešimai, tinklaraščiai, įkeltos nuotraukos, garso failai). Papildomas turinys tapo ypač aktualus paplitus antrosios kartos saityno (Web 2.0) technologijoms. Paprastai lokalizuojamas pagrindinis turinys, o papildomas turinys kuriamas konkrečios lokalios bendruomenės, kuriai skiriama svetainė.

Jei kuri nors iš aptartų svetainės dalių yra ne visiškai lokalizuota, svetainės lokalizavimas negali būti laikomas išbaigtu. Pavyzdžiui, turinio valdymo sistemos nelokalizuoti teksto pranešimai ir neadaptuoti kultūros elementai, dinamiškai patekę į tinklalapius, formuoja iš dalies lokalizuotos, iš dalies nelokalizuotos svetainės vaizdą.

Kaip ir programinės įrangos lokalizavimo atveju, svetainės lokalizavimo rezultatas priklauso nuo svetainės projektavimo. Esselink (Esselink, 2000) išskiria šiuos daugiakalbių svetainių internacionalizavimo ir lokalizavimo planavimo etapus:

1. Pasirinkti internacionalizuotą (pritaikytą daugiakalbėms svetainėms kurti) turinio valdymo sistemą, serverį ir kitą svetainei operuoti reikalingą programinę įrangą.
2. Įvertinti svetainės architektūrą ir nustatyti, kas iš svetainei sukurti pasirinktos programinės įrangos turi būti perprogramuota, adaptuota.
3. Nustatyti, kurias svetainės dalis galima išversti, o kurias iš naujo parašyti taip, kad atitiktų svetainės lankytojų kultūrinę terpę.
4. Sudaryti daugiakalbės svetainės struktūros ir turinio duomenų bazę bei naršymo sistemą.
5. Pasirinkti turinio valdymo sistemos ir turinio lokalizavimo priemones (pvz., dalinio automatizavimo priemonę, versijų sekimo priemonę, vertimo atmintį) (išsamiau apie lokalizavimo priemones žr. 7 skyrių).
6. Sukurti svetainės priežiūros ir atnaujinimo planą.

Išskiriamos trys svetainės turinio lokalizavimo tvarkymo strategijos (Sandrini, 2005):

1. **Monarchinė**, kai svetainės turinys valdomas ir lokalizuojamas centralizuotai, tačiau retai atnaujinamas. Šios strategijos taikymo pasekmė paprastai yra nejautri lokalioms rinkoms svetainė.
2. **Anarchinė**, kai didžioji svetainė turi daug paskirstytų lokalių svetainių be griežto koordinavimo, kiekviena mažesnioji svetainė turi savo dizainą. Šios strategijos taikymo pasekmė paprastai yra didelės išlaidos ir korporacijos bendros strategijos nebuvimas.
3. **Federalinė**, kuri gali būti laikoma kompromisu tarp pirmųjų dviejų strategijų, kadangi integruoja tarptautinį, regioninį ir lokalųjį turinį. Tarptautinis turinys yra kuriamas centralizuotai, lokalizuojamas ir naudojamas tarptautiniu mastu. Regioninis turinys naudojamas regiono kontekste, o lokalusis turinys kuriamas lokalės kalba, jo nelokalizuojant.

Monarchinė strategija dažniausiai taikoma dvikalbėms svetainėms arba centralizuotų tarptautinių organizacijų svetainėms. Anarchinės strategijos atveju lokalizavimas yra minimalus: visos pagrindinės lokalių svetainių dalys kuriamos lokaliai ir nepriklausomai. Esant federalinei strategijai svetainės turinio atnaujinimas turi būti integruotas į turinio publikavimo ciklą.

9. KULTŪRINĖS DIMENSIJOS

Kultūra – socialinė sistema, kurios funkcionavimas užtikrina materialinių ir dvasinių vertybių kūrimą, naudojimą ir perteikimą, apibūdina visuomenės išsivystymo lygį. Kultūra visuomet suprantama kaip stipri žmogaus aktyvaus santykio su gamta ir visuomene sąsaja, žmogaus kūrybinių jėgų ir gebėjimų plėtotė. Kultūros sąvoka labai plati, apimanti žmogaus kuriamas materialines gėrybes, žmonių dvasinės veiklos ir bendravimo produktus, priemones (mokslas, menas, filosofija, religija, pasaulėžiūra, kalba), socialinius santykius (moralė, teisė, politika), gyvenseną (papročius, tradicijas, įpročius). Visų epochų iškiliausi filosofai, mokslininkai siedavo kultūros reiškinius su tuo metu vyraujančiais gamybiniais, socialiniais, dvasiniais žmonių santykiais. Suprantama, kad šiandien kultūra neatsiejama nuo kompiuterių, modernių technologijų, todėl kuriant ar lokalizuojant programinę įrangą reikia atsižvelgti į kultūrines reikmes.

Kai kalbama apie programų lokalizavimą ar internacionalizavimą, pagrindinės kultūros problemos siejamos su žmonių dvasine veikla, bendravimu, gyvensena: kalba, religija, pasaulėžiūra, papročiais, konkrečioje aplinkoje vyraujančiais supratimo veiksniais ir pan. Kalba yra svarbiausia žmonių bendravimo priemonė, tad atsiradus kompiuterinei technikai tam iš karto buvo skiriama nemažai dėmesio ir daug nuveikta, išspręstos esminės kodavimo, suderinamumo problemos.

Pasaulyje esama daugybė įvairių kalbų (ISO 639-3 standarte apibrėžti 7299 kalbų kodai). Kalbant apie lokalizavimą ir internacionalizavimą, būtina įžvelgti esminius kalbų skirtumus. Kalbų skirtumai ypač svarbūs programinės įrangos lokalizavimui, 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 (Sapir, 1949), ryšys tarp kalbos ir kultūros yra stiprus ir daugiamatis. Todėl negalime būti tikri, kad netgi tą pačią kalbą vartojantys skirtingų kultūrų žmonės supras išsakytą arba užrašytą informaciją vienareikšmiškai.

Evers (Evers, 2001b) tyrimas parodo, kad vienoje kultūroje naudojami tos pačios kalbos išsireiškimai arba metaforos gali būti klaidingai suprantami kitoje kultūroje. Klaidingai gali būti interpretuojami ne tik kultūriškai priklausomi posakiai arba metaforos, tačiau ir daugeliu požiūrių įprasta informacija. Tam įtakos gali turėti įvairios žmonių grupių 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).

Be kalbos, kultūrai svarbūs ir kiti elementai, kurie pasireiškia daugelyje sričių naudojant informacines ir komunikacines technologijas, pavyzdžiui, rašant elektroninius laiškus, naršant internete, atliekant užduotis virtualiosiose mokymosi aplinkose, sudarant tvarkaraščius, kuriant svetaines. Daugelis kultūrinių elementų kol kas nėra vienareikšmiškai nusakomi, nėra bendrų susitarimų, standartų, dėl to sunku juos įgyvendinti. Pagrindiniai kultūros elementai nusakomi kiekvienos valstybės lokalėje: čia apibrėžiama kalba, jos abėcėlė, papildomi ženklai, šalis, žodžių rikiavimas, kalendorius, datos, skaičių formatai, matavimo vienetai. Apie lokalės sampratą, struktūrą, ypatumus dėstoma 4 skyriuje, ten išsamiai nagrinėjami lokalėse apibrėžiami pagrindiniai kultūros elementai, jų mes čia nekartosime, o aprašysime tik bendriausius kultūros ir informacinių technologijų sąryšius, pateiksime kultūrinių dimensijų modelį.

9.1. Kultūrinių dimensijų modeliai

Pasaulyje ne tik esama daugybės skirtingų kalbų ir dialektų, bet ir susiformavę daugybė įvairių papročių, požiūrių, įvairuoja socialinė elgsena. Visa tai tiria antropologai, sociologai, psichologai. Jie suformavo įvairius kultūrų modelius, kurių dimensijos leidžia aprašyti esamus kultūrų skirtumus.

Nagrinėjant visuomenės santykius dažniausiai išskiriamos keturios dimensijos[1]. Iš esmės jos atspindi visuomenės pilietiškumo raidą, tačiau jas galime taikyti ir požiūriui į visą visuomenę susidaryti. Tai – politinė, socialinė, kultūrinė ir ekonominė dimensijos. Būtina pabrėžti, kad visos keturios dimensijos vienodai svarbios ir lygia-

[1] R. Veldhuis (1997) Education for Democratic Citizenship: Dimensions of Citizenship, Core Competencies, Variables and International Activities. Strasbourg, Council of Europe, document DECS/CIT (97) 23.

vertės visuomenėje: jų reikšmingumas vaizdžiai lyginamas su keturių kojų kėde – jei bent viena koja trumpesnė, gaunama nestabilumo situacija, kyla problemų.

Kultūrinė dimensija apima bendrą įvairių šalių kultūrinį paveldą, jo suvokimą, jai priklauso ir pradinių įgūdžių – kalbos, skaitymo, rašymo – įvaldymas. Kalbant apie kultūrą, svarbu gerų pavyzdžių, etiketo, elgesio normų ir kt. propagavimas, sklaida. Tam ypatingai gerai tinka informacinės technologijos.

Vienas dažniausiai cituojamų antropologų yra olandų mokslininkas Geert Hofstede, kuris tyrinėdamas kultūrų skirtumus, atsižvelgdamas į įvairių kultūrų žmonių mąstymo, pojūčių ir veiklos šablonus, pasiūlė penkių kultūrinių dimensijų modelį (Hofstede, 1991).Vėliau šį modelį papildė ir patikslino drauge su savo sūnumi Geert Jan Hofstede (Hofstede G., Hofstede G. J., 2005). Modelio autorius išnagrinėjo medžiagą apie 53 valstybes (apklausė 116 000 įvairių profesijų žmonių), susistemino ir apibendrino jų kultūrinius skirtumus, ypatumus, pagrindinius įtakos veiksnius ir apskaičiavo jų dimensijų lygius, atspindinčius kultūrinių skirtumų šaknis. Buvo paskelbtas kultūrinių dimensijų modelis, kurį sudaro penki lygmenys – dimensijos:

1. *Galios santykis*. Nurodoma, kiek svarbi ir aiški yra hierarchija visuomenėje, koks žmonių diferenciacijos laipsnis. Ši dimensija daugiau išreikšta Lotynų Amerikos, Azijos ir Afrikos valstybėse, mažiau – Vakarų Europos valstybėse. Tvirtinama, kad vienas didžiausių galios santykių būdingas Malaizijai ir Singapūrui.
2. *Individualizmas – kolektyvizmas*. Individualių arba kolektyvinių pasiekimų vertinimas. Individualistinės kultūros vertina asmeninį laiką, laisvę, privatumą, tuo tarpu kolektyvizmas nusako stiprų priklausymą grupėms nuo pat gimimo, aukštai vertinama grupių gerovė, grupės nuomonė. Individualizmas labiau pasireiškia išsivysčiusiose, ekonomiškai ir technologiškai stipriose valstybėse, pavyzdžiui, individualizmas būdingas visoms Vakarų Europos valstybėms. Kolektyvizmas juntamas daugiau besivystančiose, ekonomiškai ir technologiškai silpnesnėse valstybėse, pavyzdžiui, tai būdinga didelei daliai Rytų Europos valstybių. Sakoma, kad Japonijoje vyrauja vidutinio individualizmo lygis, JAV – vienas aukščiausių, tuo tarpu Lotynų Amerikos šiaurės vakarų valstybės, taip pat Pakistanas, Indonezija turi labai žemą individualizmo poreikį.
3. *Vyriškumas – moteriškumas*. Atspindi lyčių vaidmenis visuomenėje. Aukštesnį vyriškumo lygį turinčiose kultūrose moterys ir vyrai turi aiškiai paskirstytus vaidmenis, tuo tarpu žemesnį vyriškumo lygį turinčiose kultūrose moterų ir vyrų vaidmenys yra panašūs. Pavyzdžiui, aukščiausią vyriškumo

lygį turi Japonija, o žemiausią – Švedija. Gana aukštu vyriškumo lygiu pasižymi Vokietija, Austrija.

4. *Neapibrėžtumo vengimas*. Atspindi, kiek visuomenė linkusi vengti neapibrėžtumo, dviprasmybės. Aukštesnio neaprėžtumo vengimo lygio kultūrose stengiamasi aiškiai apibrėžti elgesio taisykles, įstatymus. Žemesnio neapibrėžtumo kultūrose labiau toleruojama įvairovė, pokyčiai ir eksperimentavimas. Neapibrėžtumo vengimo lygis yra aukštas Lotynų Amerikos valstybėse, Japonijoje ir vokiškai kalbančiose valstybėse. Žemesnis – Šiaurės šalių, britų, Kinijos kultūrose.
5. *Ilgalaikė perspektyva* (lyginant su trumpalaike). Atspindi visuomenės „laiko horizontą“, praeities, dabarties ir ateities svarbą. Ilgalaikės perspektyvos kultūrose labiau vertinamas taupumas, ištvermingumas, atkaklumas, pragmatizmas (aukščiausius įvertinimus gavo Kinija, Japonija, Taivanis, Šiaurės Korėja). Žemiausias ilgalaikės perspektyvos lygis buvo nustatytas besivystančiose valstybėse (Pakistanas) ir dalyje Vakarų valstybių, taip pat JAV.

Nepaisant to, kad Hofstede darbai ir jų taikymas informacinių technologijų sričiai sulaukė nemažos kritikos, jo klasifikacija vis tiktai lieka populiariausia. Kultūrinės orientacijos yra giliai įsitvirtinusios kultūrose, tad modernios technologijos negali jų panaikinti, jos keičiasi labai lėtai.

Be Hofstede penkių kultūrinių dimensijų modelio buvo pasiūlyta ir kitų modelių, pavyzdžiui, septynių dimensijų modeliai (Schwartz, 1994; Trompenaars, 1997), šešių dimensijų modelis (Singh, Zhao, Hu, 2003), keturių dimensijų modelis (Hall, Hall, 1990). Kultūrinių dimensijų modeliai panagrinėti ir palyginti straipsnyje (Guseva, 2009).

Kultūrinės dimensijos svarbios lokalizuojant programas – reikia kreipti dėmesį ne tik į pagrindinius (formaliai ar neformaliai išreikštus) lokalės elementus, bet ir į gilesnius kultūrinius skirtumus. Pavyzdžiui, lokalizuojant programas reikėtų atsižvelgti į šalių kultūrines dimensijas ir, jei jos ryškiai skiriasi, išsamiau patyrinėti lokalizuojamą programą, parinkti kitokią grafinę medžiagą, sukurti kitokią struktūrą, t. y. labiau priimtiną paskirties kultūroje. Tam tikrais atvejais reikėtų net iš naujo suprojektuoti kai kurias programinės įrangos funkcionalumo dalis ar net perprogramuoti grafinę sąsają. Tai dar kartą patvirtina, kad turi vykti glaudus lokalizuotojų ir programų projektuotojų bendradarbiavimas.

Ryškūs Rytų ir Vakarų kultūriniai skirtumai, partikuliarizmo ir universalizmo, individualizmo ir kolektyvizmo vyravimas, skirtingos kultūrinės vertybės daro poveikį programinės įrangos projektavimui ir naudojimui (Hall, Hall, 1990; Trompenaars, 1997; Lanier, 2005). Keliama idėja, kad ne visada ir ne visoms kultūroms įmanoma

kokybiškai lokalizuoti bet kurią kompiuterio programą, – dėl to kartais verta kurti atskiras programų versijas, specialiai skirtas toms kultūroms.

Dabartinių lokalizacijų kokybę mažina tai, kad šiuo metu dauguma lokalizavimo sprendimų priima ne konkrečios kultūros (kuriai lokalizuojama) atstovai, o programinės įrangos projektuotojai (pvz., dirbantys Jungtinėse Amerikos Valstijose). Dėl šių priežasčių atvirosios programos yra lankstesnės, nes lokalizuojant galima suprojektuoti iš naujo arba pakoreguoti neatitinkančias lokalei dalis. Pavyzdžiui, nustatyta, kad „Microsoft“ korporacijos operacinės sistemos nėra skirtos kolektyvinėms nuostatoms įgyvendinti – naudotojų profiliai pabrėžia individualumą. Kai kurie programinės įrangos kūrimo specialistai siūlo teikti nebaigtus kurti produktus, kad juos būtų galima baigti projektuoti ir lokalizuoti konkrečiose valstybėse.

Siūlomos idėjos sukurti tokią programinę įrangą, kuri savaime prisitaikytų prie naudotojo kultūrinių normų (Reinecke, Bernstein, 2007). Pasinaudojant minėtomis kultūrinėmis dimensijomis sukuriamos kultūrinės žymės – programinės įrangos sąsajos elementai, pageidautini tam tikroje kultūroje, naudojami modeliavimo metodai ir sąsajos pritaikymo pasiekimai iš dirbtinio intelekto sričių. Pagrindu laikoma asmens valstybė ir kalba, o visi kiti elementai modeliuojami naudojant stereotipus, bendruomenės nuostatas, analizuojant žmogaus sąveiką su sistema. Kol kas sklando tik idėjos, realizacijos dar nėra.

Programinės įrangos gamintojai, norėdami sumažinti programinės įrangos, skirtos lokalizavimui, tekstų apimtis, dažnai pakeičia juos vaizdine (grafine) informacija, pavyzdžiui, piktogramomis, paveikslais. Iki šiol tebevyrauja klaidinga nuomonė, jog grafiškai pateiktos informacijos nereikia lokalizuoti, tačiau, kaip matėme iš anksčiau pateiktų samprotavimų, jei valstybių kultūrinės dimensijos stipriai skiriasi, tai būtina lokalizuoti visą vaizdinę informaciją, net jeigu joje ir nėra teksto. Pastebėsime, kad lokalizuoti reikia ne tik lingvistinei aplinkai (kaip jau yra įprasta), bet ir bendrijoms, kurių narius jungia kiti požymiai, pvz., amžius (Hoffman, 2007). Amžiaus grupėms yra ypač svarbu parinkti tinkamas piktogramas ir kitus sąsajos elementus.

Viename iš atliktų programinės įrangos vizualizavimo tyrimų dalyvavo respondentai iš JAV, Vokietijos ir Kinijos (Auer, Dick, 2007). Kiekvienam dalyviui buvo pateiktas tradicinių piktogramų rinkiniai, iš kurių jis turėjo parinkti jo manymu tinkamą piktogramą nurodytai funkcijai atlikti. Eksperimentui buvo parinktos 28 funkcijos. Tyrimu nustatyta, kad esama piktogramų interpretavimo skirtumų, priklausančių nuo respondentų valstybės: respondentai iš Kinijos kiek kitaip interpretavo pateiktas programas ir atpažino mažiau piktogramų. Buvo naudojamos ir nuo kultūros priklausomos piktogramos, ir nepriklausomos. Tai reiškia, kad JAV kuriamas

piktogramas nevienodai interpretuoja Kinijos gyventojai ir, būtent, tai reikia numatyti projektuojant programinę įrangą.

Evers daktaro disertacijos nuotolinio mokymo internetinės aplinkos tyrimu (Evers, 2001a) parodyta, kad metaforiškai pateikiami programinės įrangos sąsajos elementai skirtingų kultūrų atstovams kelia skirtingas asociacijas, o tą patį tikslą pasiekti dažnai naudojami skirtingi veiksmai. Tą patį patvirtina ir elektroninio mokymosi svetainių lokalizavimo tyrimas (Mushtaha, De Troyer, 2005). Tyrime dalyvavo studentai iš Palestinos ir Belgijos, kurie reguliariai naudojosi elektroninio mokymosi svetainėmis. Viena iš užduočių buvo, pasižiūrėjus į piktogramą, parašyti, kokia galėtų būti jos reikšmė ir paskirtis. Rezultatai parodė, kad piktogramos ne visada buvo teisingai interpretuojamos. Tačiau nebuvo labai žymaus klaidų skirtumo tarp belgų ir palestiniečių atsakymų, o tai reiškia, kad jauni žmonės, kurie intensyviai naudoja internetą ir elektroninio mokymosi svetaines, įgyja naujų kultūrinių verčių, keičia, plečia savo nuostatas. Manoma, kad kai kurie kultūriniai skirtumai laikui bėgant gali išnykti.

9.2. Kultūros elementų klasifikacija

Mokslininkai bando aprašyti kultūros elementus, juos klasifikuoti, sisteminti. Šiame dešimtmetyje įsivyravus holistiniam požiūriui į daugelį dalykų, atsirado ir holistinis požiūris į kultūrą. Teigiama, kad vienos ar kitos šalies kultūra yra vientisa, kad nėra nei universalių simbolių, nei universalių dėsnių – kiekviena teritorija, žmonių bendruomenė turi savo unikalią ženklų kalbą plačiąja prasme (Kersten, 2002). Tuo tarpu kiti mokslininkai palaiko redukcionistinę kultūros sampratą, kai kultūra laikoma kalba, kurios ženklus galima lengvai interpretuoti, perkelti iš vienos teritorijos į kitą, taip pat teigiama, kad galima sukurti universalių dėsnių rinkinius, schemas, kurios lengvai prigytų įvairiose kultūrose. Redukcionistinio požiūrio ištakos glūdi amerikiečio sociologo ir antropologo Kuperio veikaluose (Kuper, 1999).

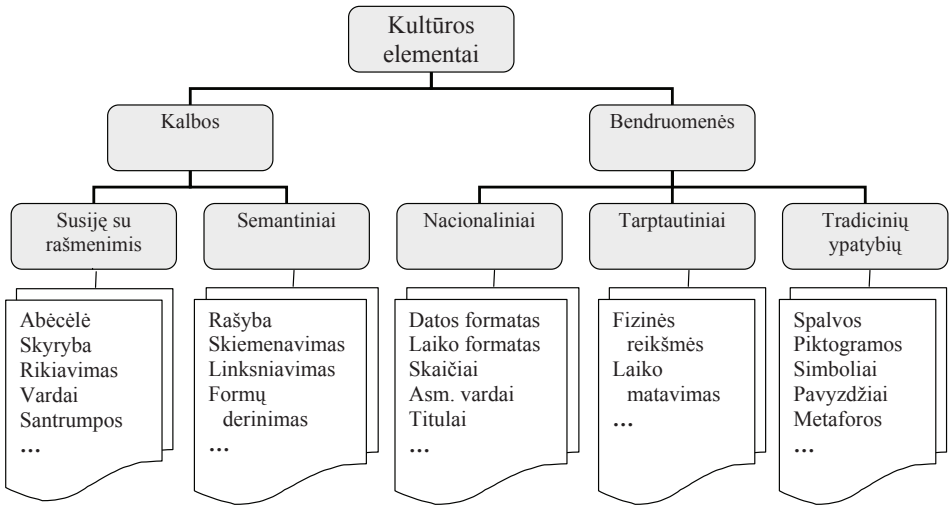
Kalbėdami apie kultūrą programinėje įrangoje, norėdami lokalizuoti arba internacionalizuoti programas, turime nagrinėti kultūros atspindžius joje – konkrečius programinės įrangos elementus.

Kaip jau minėjome, kultūros elementų, įtrauktų į lokalių modelius, nepakanka norint visiškai lokalizuoti programinę įrangą. Nemažai tyrimų (Chavan, 2007; Reinecke, Bernstein, 2007; Ford, Gelderblom, 2003; Chen, Tsai, 2007; Batra, Bishu,

2007 ir kt.) patvirtina sunkiau formalizuojamų, gilesnių kultūrinių dimensijų svarbą projektuojant ir lokalizuojant programinę įrangą, ypač skirtą internetui.

Toliau pateikiami kultūros elementai išskirti remiantis anksčiau aprašytais lokalių modeliais ir atitinkamais standartais (4.3 skyr.), taip pat ilgamete šios knygos autorių programinės įrangos lokalizavimo patirtimi. Autoriai yra lokalizavę ir išsamiai ištyrę šias programas ar jų grupes: 1) naršykles („Mozilla Firefox“, paketo „Mozilla“ naršyklę, „Opera“); 2) elektroninio pašto programas („Mozilla Thunderbird“, paketo „Mozilla“ elektroninio pašto programą, „Operos“ naršyklės pašto komponentą); 3) virtualiąją mokymosi aplinką „Moodle“; 4) turinio valdymo sistemas ir jų pagrindų sukurtas mokymo turinio tvarkymo bendradarbiaujant sistemas („Plone“, „Lemill“); 5) pokalbių programą „Chatzilla“; 6) kalendoriaus ir laiko planavimo programas („Mozilla Calendar“, „Sunbird“); 7) raštinės programų paketą „OpenOffice.org“; 8) tekstų rašyklę „Abiword“ 9) komponentinio programavimo sistemą „BlackBox“ ir kt. Pagrindiniai kultūrines dimensijas atspindintys rezultatai pateikti straipsnyje (Dagienė, Jevsikova, 2009).

Kultūros elementus galima sąlyginai suskirstyti į kalbos ir bendruomenės grupes (58 pav.). Kalbos grupės elementai gali būti susiję su rašymo sistema (pvz., abėcėle): rašmenys, skyryba, rikiavimas, vardai ir rašmenų vartojimas varduose, santrumpų vartojimas; arba priklausyti nuo kalbos semantikos: rašyba, skiemenavimas, linksniavimas, formų derinimas, vienos frazės komponavimas iš kelių. Bendruomenės grupės elementai ne tiek priklauso nuo kalbos, kiek nuo kitų susitarimų ir tradicijų,



58 pav. Kultūros elementų programinėje įrangoje struktūra

kurios gali būti priimtos nacionaliniu (valstybės) lygiu (pvz., datos, laiko, asmenų vardų, titulų rašymo taisyklės), tarptautiniu lygiu (pvz., laiko matavimas, matavimo vienetai, fizinės reikšmės) arba priklausyti nuo gilių ir sunkiau vienareikšmiškai nusakomų tradicijų (spalvų, piktogramų, metaforų ir kt. vartojimo).

Nuo gilių įvairių kalbų skirtumų priklausantys kalbos elementai nėra įtraukti į šiuolaikinius lokalių modelius ir standartus dėl sudėtingo jų formalizavimo. Dar kiti elementai, nors ir įtraukti į lokalės formaliuosius aprašus, nėra naudojami programinės įrangos tam tikrose vietose. Pavyzdžiui, nepaisant to, kad kalbos naudojama ženklų koduotė yra įtraukta į formalų lokalės aprašą (kiekviename iš egzistuojančių lokalių modelių, žr. 4.3 skyr.), galime pastebėti, kad nacionalinių kalbos ženklų paprastai nepaisoma įvairių programos objektų varduose (failai, aplankai, interneto sritys, prisijungimo vardai, slaptažodžiai ir kt. (Jevsikova, 2006)). Kita problematiška sritis – tai semantiškai išreiškiami kalbos elementai, pavyzdžiui, žodžių gramatinių formų derinimas, tekstų komponavimas atsižvelgiant į kontekstą.

Kai kurių dažniausiai naudojamų lokalės elementų reikšmės paprastai būna įtrauktos į operacinę sistemą (ženklų koduotė, dešimtainės trupmenos skirtukas, tūkstančių skirtukas, datos ir laiko formatas, pinigų vienetų formatas ir t. t.). Internacionalizuota programa turėtų šiuos elementus aptikti iš operacinės sistemos nuostatų ir juos naudoti. Priešingu atveju turėtų būti galimybė nurodyti su lokale susijusias numatytąsias nuostatas programos lokalizavimo metu, t. y. įtraukti į lokalizuojamąją programos dalį.

Kurios nors su lokalizavimu susijusios programos nuostatos nebuvimas arba jos neįtraukimas į lokalizuojamuosius išteklius laikomas internacionalizavimo klaida.

Piktograma yra labai paplitęs naudotojo sąsajos elementas. Dažnas jų naudojimas remiasi tuo, kad didina vartotojo produktyvumą, nes yra greičiau atpažįstamos, negu tekstas. Vienas iš piktogramų nepatogumų – jos gali būti klaidingai suprastos. Piktograma bus naudojama teisingiau ir tiksliau, jei meniu juostoje ji bus derinama su tekstu.

9.3. Kultūriniai elementai tinklalapiuose

Didėjant informacinių technologijų įtakai visose sferose, kalbama apie interneto galimybes pasiekti bet kurios valstybės rinką. Kadangi informacijos plitimas daugiausiai vyksta naudojantis portalais, svetainėmis ar tinklalapiais, tai susirūpinta jų lokalizavimu ir internacionalizavimu. Būtina tinkamai lokalizuoti tinklalapius skirtingų kultūrų naudotojams. Atsirado net šūkis: mąstyti globaliai, veikti lokaliai.

Pasiremdami kultūrinių dimensijų modeliais, Wan Mohd Isa ir Md Noor sukūrė rekomendacijas interneto svetainių naršymo schemai ir turiniui projektuoti (Wan Mohd Isa, Md Noor, 2007). Pavyzdžiui, svetainėse, skirtose kultūroms, kurių aukštas neapibrėžtumo vengimo lygis, reikėtų naudoti ribotą spalvų skaičių, nepiktnaudžiauti garso įrašais, informaciją dėstyti dalimis pagal temas, įtraukti su šalies tradicijomis susijusias temas, klientų konsultavimo paslaugą, platinamų produktų demonstracinių versijų parsisiuntimą bandymui, mažinti naudotojų galimų klaidų skaičių. Tuo tarpu, jei svetainė skirta žemo neapibrėžtumo vengimo lygio kultūros atstovams, tai informaciją derėtų dėstyti pagal užduotis, informaciją perteikti įvairiais būdais: spalvomis, garso įrašais, animacija.

Svetainėse, skirtose aukšto (sodraus) konteksto kultūrai, informacija yra nuspėjama, netiesioginė, suprantama iš konteksto, todėl dėmesys telkiamas estetinėms detalėms, laisvam spalvų naudojimui, akcentuojamas vaizdas ir kontekstas. Žemo (skurdaus) konteksto kultūrų svetainėse dominuoja informatyvesnė, tiesioginė ir mažiau nuspėjama informacija, todėl nuorodos išdėstomos abėcėlės tvarka, naudojamas tikslus ir aiškus puslapio išdėstymas.

Vertinant kultūrinių skirtumų įtaką tinklalapių turiniui, galima išskirti keletą lygmenų, dažniausiai nurodomi trys lygmenys (Singh, 2002):

1) Suvokimo lygmuo, kai dėmesys skiriamas kalbai ir spalvoms;
2) Elgsenos lygmuo, kai dėmesys skiriamas kultūrinėms vertybėms;
3) Simbolinis lygmuo, kai dėmesys skiriamas visuomenės ženklams ir simboliams suprasti.

Suvokimo lygmenyje N. Singh tinklalapiams analizuoti siūlo išskirti keletą vertinimo rodiklių, pavyzdžiui: erdvinę orientaciją, tinklalapio struktūrą, teksto dydį, navigacijos galimybes, vertimo atitikmenis, dialektus, spalvų kategorijas.

Erdvinė orientacija reiškia tinklalapio turinio išdėstymą ekrane. Čia reikia atsiminti, kad vienų kalbų tekstai skaitomi vertikaliai, kitų – iš kairės į dešinę arba atvirkščiai. Verčiant tinklalapį būtina į tai atsižvelgti.

Tinklalapio struktūra turi išryškinti svarbiausias vietas pateikiant jas matomiausioje vietoje, kas priklauso kultūrinėms dimensijoms. Svarbu kreipti dėmesį į didžiųjų raidžių rašymą, jų suvokimą tam tikrose kultūrose. Raidžių ar žodžių pabraukimas, šriftų rūšys, stiliai taip pat turi svarbos, ypač kai yra ryškūs kultūriniai skirtumai.

Teksto dydis susijęs su tinklalapio ilgumu, taip pat grafikos, lentelių, linijų naudojimu.

Navigacija tinklalapyje – svarbus teksto skaitymo atributas. Žmonės mėgsta naršyti ta tvarka, kuria jie ir rašo, pavyzdžiui, iš kairės į dešinę, iš dešinės į kairę ar vertikaliai.

Vertimo atitikmenys reikalingi, jei iš pat pradžių manoma, kad tinklalapis bus verčiamas į kelias kalbas. Tuomet reikia kruopščiai pasirūpinti žodynu ir, žinoma, atidžiu lokalizuoto tinklalapio testavimu.

Dialektai svarbūs toms šalims, kuriose vartojami skirtingi tos pačios kalbos dialektai. Pavyzdžiui, nors Portugalijoje ir Brazilijoje kalbama ta pačia portugalų kalba, tačiau skirtumų esama, ir nemenkų.

Spalvų kategorijos. Tai viena subtiliausių kultūrinės dimensijos charakteristikų. Skirtingos tautos emocijas išreiškia skirtingomis spalvomis. Ne visur liūdesys nusakomas juoda spalva. Neretai pasitaiko, kad spalvinis tinklalapio nepriimtinumas atsiranda dėl dizainerio klaidų ar blogo komponavimo, o ne dėl kultūrinių skirtumų.

Irina Kondratova ir Ilia Goldfarb (2006) atliko tyrimą, kurio metu išanalizavo penkiolikos valstybių populiariausias svetaines naudojamų spalvų atžvilgiu. Remiantis rezultatais kiekvienai valstybei buvo išskirtos šešiolika spalvų, kurios buvo dažniausiai naudojamos. Sudarius bendrą sąrašą paaiškėjo, kad pirmos dešimt spalvų dominuoja kiekvienoje iš penkiolikos valstybių. Tai balta, skirtingų atspalvių juoda, pilka, mėlyna ir šviesiai geltona spalva. Šių spalvų paletę galima pavadinti „tarptautine spalvų palete“. Tyrėjai mano, kad spalvas iš šios paletės galėtų naudoti svetainių kūrėjai, tai pagerintų svetainių lankomumą.

Mokslininkai, analizuodami kultūrinių dimensijų įtaką svetainėms ir portalams, yra parengę klausimų sąrašą, į kuriuos vertėtų atsakyti prieš projektuojant arba lokalizuojant produktą tam tikrai kultūrai (Marcus, Gould, 2001). Keletas iškylančių klausimų: Ar tokie veiksniai, kaip nuoširdumas ir sąžiningumas, turi būti naudojami kreipiniuose? Ar tai darys įtakos neigiamam elgesiui? Ar išsikeltas tikslas mokytis santūrumo gali pakeisti tai, kas jau buvo išmokta individualizmo arba kolektyvizmo kultūrose? Ar individualizmo ir kolektyvizmo kultūrų svetainės turi būti kuriamos ir naudojamos skirtingai? Ar turi būti skirtingų svetainių, skirtų vyrams ir moterims priklausomai nuo jų šalies ar religijos?

Tinklalapiams kurti naudojamos turinio valdymo sistemos, skirtos įvairioms kalboms. Deja, dabartinės turinio valdymo sistemos neapima daugumos kultūrinių aspektų. Šiuo metu norint adaptuoti svetainę konkrečiai kultūrai, reikia skirti daug laiko ir lėšų: reikia ne tik taisyklingai išversti, pakeisti matavimo vienetus ir kita, bet ir rasti kultūros ekspertą, kuris įvertintų atitikimą vienai ar kitai šaliai.

Informacija, pateikiama tinklalapiuose, gali būti elektroninio verslo sėkmės pradžia. Todėl būtina apibrėžti tikslinę grupę ir pritaikyti tinklalapį pagal grupės poreikius. Lokalizavimas kultūrai turi didelį potencialą gerinant tinklalapių, svetainių, portalų kokybę ir efektyvumą.

Pabaigos žodis

Knygoje dėstoma medžiaga atspindi dabartinę programinės įrangos lokalizavimo situaciją, apžvelgiamas ir nueitas kelias. Mums aktualus programinės įrangos lietuvinimas nagrinėjamas ne izoliuotai, o kitų kalbų, ypač vartojančių lotynų rašmenis, kontekste, lyginant lietuvių kalbos ypatybes su kitomis kalbomis. Daugelis problemų ir jų sprendimų, atsiradusių lietuvinant programinę įrangą, yra bendri ir kitoms panašios struktūros (sintetinėms, fleksinėms) kalboms.

Lokalizavimo darbai Lietuvoje pradėti dešimtmečiu vėliau, negu Vakarų Europoje, bet šį tą jau turime. Parašyta kelios dešimtys mokslinių straipsnių, apgintos dvi disertacijos (Laucius, 2007; Jevsikova, 2009). Šių darbų tikslas – mažinti lokalizavimo darbų sąnaudas ir gerinti lokalizacijų kokybę.

Viena iš efektyviausių lokalizuojamųjų išteklių vertimą automatizuojančių priemonių yra vertimo atmintis. Norėdami ja pasinaudoti, turime turėti teksto segmentų žodyną. Kol kas atskiri programų lietuvintojai juos daro atskiroms programoms. Norint efektyviai (kad nereikėtų kiekvienam kurti savo duomenų bazės) pasinaudoti vertimo atmintimi ir pasiekti aukštesnės lokalizavimo kokybės, reikalinga norminių lietuviškų teksto segmentų bazė. Šiai bazei sukurti būtinas žmogaus ir kompiuterio sąsajos leksikos žodynas. Tokio žodyno prototipą jau turime (Dagienė ir kt., 2008), bet segmentų bazės – dar ne. Ją sukurti būtų artimiausios ateities darbas.

Daugeliu programų (operacine sistema, tekstų rengykle, interneto naršykle, elektroniniu paštu, pokalbių programa ir kt.) žmogus naudojasi beveik kasdien ir mato tuos pačius tekstus, rodomus kompiuterio ekrane. Nuolat matomų tekstų poveikis žmogaus kalbinės kultūros formavimuisi kur kas didesnis, negu skaitomų knygų ar žurnalų, todėl šiems tekstams reikia skirti ypatingą dėmesį. Tai gali būti pasiekta glaudžiau bendradarbiaujant kalbininkams ir informatikams.

Programų sąsajos tekstų vertimas sudėtingesnis negu ištisinių tekstų, jis priklauso nuo konteksto, kurio vertėjas paprastai nemato. Būtina tobulinti kompiuterinio (mašininio) vertimo priemones, kad jos geriau atsižvelgtų į kontekstinę informaciją, ją suformuotų, apdorotų ir perduotų vertėjui. Vienas iš konteksto formalizavimo būdų – panaudoti atributines gramatikas.

Būtų gerai, kad konteksto formalizavimą atliktų originalios programos autorius kaip vieną iš programos internacionalizavimo darbų. Daugumos originalių programų sąsaja rašoma anglų kalba, kuri sunkiai formalizuojama dėl nevienareikšmiškumų ir homonimų gausos. Vertas dėmesio P. A. V. Hall ir R. Hudson (1997) pasiūlytas vertimo problemų sprendimo kelias – naudoti tarpinę kalbą, kuri būtų nepriklausoma nuo jokios konkrečios kalbos. Tai radikalus internacionalizavimo kelias. Jis vertingas tuo, kad taip internacionalizavus originalią programą iš esmės palengvėtų jos lokalizavimas kitoms kalboms. Lokalizuotojai gautų tikslesnę, aiškesnę ir neužterštą originalo kalbos subtilybėmis informaciją tiesiog iš išteklių eilučių.

Lietuvoje mažai kuriama programų, kurios būtų lokalizuojamos kitoms kalboms. Tačiau internacionalizavimo problemos ir klaidos mums geriau matomos, negu programų autoriams, dirbantiems vienos kalbos terpėje.

Bet kurios mokslo šakos darbams turi įtakos studijos universitetuose. Lietuvoje kol kas tėra tik viena su lokalizavimu susijusi studijų programa – technikos kalbos vertimas ir lokalizavimas (Kauno technologijos universiteto Humanitarinių mokslų fakultete). Vilniaus universiteto Matematikos ir informatikos fakultete pedagoginių studijų bakalaurams skaitomas programinės įrangos lokalizavimo kursas. Deja, būsimi programuotojai, rengiami keliuose šalies universitetuose, su internacionalizavimu ir lokalizavimu nesupažindinami.

PRIEDAI

1 priedas. Programų lokalizavimo terminų žodynėlis

akū́tas. Žr. dešininis kirtis.

alternatỹvos klavìšas. Klaviatūros klavišas, kurį paspaudus kartu su kitu klavišu, galima gauti to klavišo alternatyvią funkciją. Žymimas *Alt*.

analìtinė kalbà. Kalba, kurioje santykiai tarp žodžių išreiškiami pagalbiniais žodžiais, žodžių tvarka sakinyje dažniausiai griežta, pvz., anglų kalba. Plg. sintetinė kalba.

ANSI standártas [sk. ansi] (angl. ANSI = American Standards Institution – JAV standartizacijos institutas). JAV nacionalinis standartas.

añtrojo lýgio klavìšas. Klavišas, kurį laikant paspaustą renkami antrojo lygio (registro) ženklai: didžiosios raidės arba kiti ženklai, užrašyti klavišo viršutinėje dalyje. Lietuviškoje standartinėje klaviatūroje žymimas *Lyg2*.

apýtikslė atitiktìs. Išteklių eilutės vertimas, ne visai atitinkantis eilutę, esančią vertimo atmintyje. Išreiškiama atitikimo procentais (0–99 proc.) arba trupmeniniu, už vienetą mažesniu skaičiumi (0–0,9999).

ASCII koduõtė [sk. asky] (angl. ASCII = American Standard Code for Information Interchange – Standartinis JAV informacijos mainų kodas). Septynių bitų koduotė, įteisinta JAV standartu ir užregistruota kaip tarptautinio standarto modifikacija ISO 646-IRV.

AZERTY klaviatūrà. Klaviatūra, kurios viršutinė raidžių klavišų eilė prasideda raidėmis AZERTY. Naudojama Prancūzijoje, Belgijoje.

ĄŽERTY klaviatūrà [sk. ažerty...]. Klaviatūra, kurios viršutinė klavišų su raidėmis eilutė prasideda raidėmis ĄŽERTY.

báitų eiliškùmas. Baitų, vaizduojančių skaičiaus skaitmenis, išdėstymo seka kompiuterio atmintyje: pradedant nuo aukščiausios skaičiaus skilties arba nuo žemiausios.

báitų eiliškùmo žénklas. Ženklas, nurodantis baitų eiliškumą unikodo tekste. Kodas U+FFFE. Rašomas unikodo teksto failo pradžioje.

bãzinis žõdis. Programavimo kalbos žodis, turintis tam tikrą, toje kalboje apibrėžtą, reikšmę.

cólis. Nesisteminis ilgio vienetas, lygus 2,54 cm, vartojamas JAV.

daugiakaĺbis rikiãvimas. Daugiakalbio teksto rikiavimas pagal daugiakalbio rikiavimo taisykles. Rezultatas gali skirtis nuo vienkalbio rikiavimo.

dešinìnis kiřtis. Ženklas ´. Kodai: 180 (Windows-1257, dešimtainis), U+00B4.

diakrìtinis žénklas. Ženklelis virš raidės, po raide, greta jos arba ant jos, tą raidę modifikuojantis arba paverčiantis kita raide.

dialògo lángas. Kompiuterio ekrane rodomas langas, kuriame galima pateikti informaciją, reikalingą tolesniam sistemos darbui.

didė́jantys báitai. Unikodu koduoto ženklo baitų išdėstymo būdas, kai pirmiau eina vienetų baitas, po jo – šešioliktukų (šešioliktainės skaičiavimo sistemos) baitas. Žymimas LE, pvz., UTF-16LE. Žr. dar baitų eiliškumo ženklas. Plg. mažėjantys baitai.

dù taškaĩ viršujè. Ženklas ¨. Kodai: 141 (Windows-1257, dešimtainis), U+00A8.

dvibaĩtė koduõtė. Koduotė, kurioje vienas ženklas koduojamas dviem (arba vienu) baitais. Plg. daugiabaitė koduotė, vienbaitė koduotė.

dvikrỹptė kalbà. Kalba, kurioje rašoma abiem kryptimis: iš kairės į dešinę ir iš dešinės į kairę. Dvikryptės yra hebrajų ir arabų kalbos. Jų žodžiai rašomi iš dešinės į kairę, o tekste pasitaikantys kitų (vienkrypčių) kalbų žodžiai ir skaičiai – iš kairės į dešinę.

dviraĩdis. Dvi raidės, reiškiančios vieną garsą. Dviraidis užima atskirą vietą abėcėlėje. Kai žodžiai rikiuojami, abi raidės traktuojamos kaip vienas ženklas pagal dviraidžio užimamą vietą abėcėlėje.

EBCDIC koduõtė (sk. e-bė-cė-dik...) (angl. EBCDIC = Extended Binary Code Decimal Interchange Code – Išplėstasis dvejetainis kodas ir dešimtainis informacijos mainų kodas). Aštuonių bitų koduotė, skirta didiesiems kompiuteriams. Ją sukūrė IBM bendrovė ir dėjo į tuo metu populiarius kompiuterius „IBM System/360“.

eilùtė. Žr. išteklių eilutė.

ergonóminė klaviatūrà. Klaviatūra, kuria galima sparčiau rinkti tekstą ir renkant mažiau pavargstama. Ergonomiškumas pasiekiamas parenkant racionalų ženklų išdėstymą, pritaikytą konkrečiai kalbai.

flèksinė kalbà. Kalba, kurios žodžiai pasižymi gramatinių formų įvairove, šaknies balsių kaita.

fonètinė kalbà. Kalba, kurioje aiškus ir vienareikšmis santykis tarp raidžių ir garsų: vieną garsą atitinka viena raidė (rašto ženklas). Lietuvių kalba artima fonetinei, anglų – nefonetinė.

glìfas. Rašmens piešinys. Suprantamas nevienodai: 1) abstraktus, t. y. nepriklausomas nuo rašmens vaizdavimo šrifto arba stiliaus ir 2) konkretus, t. y. kiekvienas kitoks rašmens vaizdavimas (pvz., normaliu stiliumi arba kursyvu) yra skirtingi glifai.

globalizãcija. Programinės įrangos, interneto svetainių ir kitų elektroninių priemonių pritaikymas naudoti pasauliniu mastu.

GNU [sk: gnu] (angl. GNU = GNU's not UNIX). Laisvai platinama atviroji programinė įranga, skirta „Linux" operacinei sistemai.

grãfinė sąsaja. Grafikos priemonėmis pagrįsta ryšio ir sąveikos priemonių tarp žmogaus ir kompiuterinės programos visuma. Komandoms parinkti, programoms paleisti, failų ir aplankų vardams, parinktims, parametrams stebėti ir parinkti, taip pat kitiems veiksmams atlikti naudojami ekrane rodomi dialogo langai, meniu punktai, mygtukai ir kt. elementai, kuriais manipuliuojama pele, klaviatūra ar kt. manipuliavimo įtaisu.

gramãtikos klaidà. Dokumento teksto neatitikimas tam tikros kalbos gramatikos taisyklėms.

gramãtikos tikrintùvė. Programinės įrangos komponentas klaidingoms gramatinėms konstrukcijoms ieškoti.

gramãtinė fòrma. Žodžio forma, gauta jį kaitant: linksniuojant, asmenuojant, laipsniuojant.

grãvis. Žr. kairinis kirtis.

hieròglifas. Rašto ženklas, žymintis daiktą, jo rūšį, sąvoką arba priebalsių junginį. Hieroglifai yra egiptiečių, kinų, japonų, arabų ir kitų tautų ideografinio rašto ženklai.

hiraganà. Ideografinis skiemeninis japonų raštas, sudarytas iš hieroglifų ir kanos rašmenų.

i18n [sk: i-aštuoniolika-en]. Programuotojų žargone vartojama žodžio „internacionalizavimas" santrumpa. Šio simbolinio žymens pirmoji ir paskutinė raidės yra angliško žodžio „internacionalization" pirmoji ir paskutinė raidės, o skaičius 18 rodo, kiek šio žodžio raidžių praleista santrumpoje tarp pirmosios ir paskutinės raidės.

internacionalizãcija. 1. Programinės įrangos projektavimo proceso, pritaikant ją įvairioms kalboms ir kultūroms, rezultatas. **2.** Programinės įrangos savybių, darančių ja lengvai adaptuojama įvairioms kalboms ir kultūroms, visuma.

internacionalizãcijos klaidà. Programinės įrangos projektavimo klaida, kai tam tikra jos savybė arba parametras nenumatomas įvairioms lokalėms pritaikyti.

internacionalizãvimas. Programinės įrangos ir jos duomenų struktūrų projektavimas taip, kad visa tai būtų galima lengvai adaptuoti įvairioms kalboms ir kultūroms.

internacionalizúotasis ìštekliaus identifikãtorius. 32 bitų ISO 10646 koduotės ženklų seka, identifikuojanti interneto išteklių.

internacionalizúotasis sritiẽs vařdas. Interneto srities vardas, kurio komponentuose (hierarchijos lygį nusakančiose dalyse, atskiriamose taškais) galima vartoti unikodo ženklus.

invariántinė kultūrà. Internacionali abstrakti kultūra, nesiejama su konkrečios šalies kultūra arba kalba. Plg. neutralioji kultūra.

IPA fonètiniai ženklaĩ [sk. i-pa...]. Tarptautinio fonetikos instituto (IPA) reglamentuojami fonetiniai ženklai.

ISO standártas [sk. iso...] (gr. iso = ίσος (lygu)). Tarptautinės standartų organizacijos sukurtas standartas.

ìštekliaus identifikãtorius. Bet kokį išteklių identifikuojantis pavadinimas.

ìšteklių atskyrìmo metòdas. Veiksmų ir priemonių visuma programos ištekliams atskirti nuo vykdomosios dalies.

ìšteklių eilùtė. Lokalizuojamųjų išteklių ženklų seka, skirta rodyti ekrane (pvz., meniu punkto, komandos, klaidos pranešimo tekstas) arba nurodyti tam tikrą lokalizuojamosios programos nuostatą (pvz., koduotę, šriftą).

ìšteklių fáilas. Failas, į kurį sudėti programos ištekliai.

ìšteklių kompiliãtorius. Programa, iš tekstinio išteklių failo gaminanti dvejetainį išteklių failą arba įdedanti dvejetainę išteklių sekciją į vykdomąjį failą.

ìšteklių sèkcija. Programos modulio vykdomojo failo sekcija, į kurią sudėti programos ištekliai.

ìšteklių tvarkytùvė. Programa, skirta programų ištekliams tvarkyti: versti į kitas kalbas, langų ir laukų dydžiams keisti.

kairìnis kiřtis. Ženklas `. Kodai: 96 (ASCII, dešimtainis), U+0060.

kalbõs identifikãtorius. Konstanta, identifikuojanti kalbą, kuriai suderinta sistema arba kuria parašytas dokumentas. Gali būti skaitinis arba abėcėlinis.

kalbõs kòdas. Dviem arba trimis raidėmis pažymėtas kalbos pavadinimas, paprastai sudarytas iš tos kalbos originalaus pavadinimo raidžių.

kalbõs nòrma. Pagal nustatytas taisykles visuotinai pripažintas kalbos faktas ir jo vartojimas žodžiu ir raštu.

kalbõs núostatos. Nuostatų, priklausomų nuo kalbos, grupė. Tai pagrindiniai lokalės elementai, susiję su apdorojamų dokumentų kalba: klaviatūra, skaičių, datos, laiko, valiutos ir kiti formatai.

kalbų inžinèrija. Rašytinės ir šnekamosios kalbos žinių panaudojimas kuriant kalbos ir šnekos atpažinimo, analizės, sintezės ir vertimo kompiuterines priemones.

kandžĩ. Ideografinis japonų raštas, kurio ženklai paimti iš kinų.

katakanà. Ideografinis skiemeninis japonų raštas, sudarytas iš hieroglifų ir kanos rašmenų.

kėlìmo žénklas. Ženklas, kurį įterpia skiemenavimo programa arba rašyklės skiemenavimo komponentas į tas žodžių vietas, kuriose žodis gali būti keliamas į kitą eilutę.

kiřčio žénklas. Diakritinis ženklas, vartojamas kirčiui žymėti – kirčiuotai raidei gauti.

kirčiúota raĩdė. Raidė su kirčio ženklu. Lietuvių kalboje vartojami trys kirčio ženklai: kairinis, dešininis, riestinis kirtis.

kirìlicos rãšmenys. Kirilicos rašto kalbose (pvz., rusų, baltarusių, bulgarų, ukrainiečių) vartojamų rašmenų aibė.

klaviatū̃ros lỹgis. Klaviatūros būsena, nustatanti, kuris ženklas bus surinktas paspaudus klavišą, kai klavišas skirtas dviem arba trims ženklams rinkti.

klaviatū̃ros ženklų sritìs. Pagrindinė klaviatūros dalis, kurioje išdėstyti visi ja renkami ženklai ir dalis pagrindinių valdymo klavišų.

klavìšas. Klaviatūros, pelės arba kitokio įtaiso elementas, kurį paspaudus gaunamas ženklas arba atliekamas tam tikras veiksmas (vykdoma komanda).

klavìšo kòdas. Kodas, kurį klaviatūra siunčia į procesorių, kai paspaudžiamas arba atleidžiamas klavišas.

kodãvimas. Veiksmas, kuriuo ženklui arba kitokiam objektui suteikiamas kodas.

kòdo pãkaitalas. Vienas ar daugiau kodų, kuriais pakeičiamas ženklo kodas, kai dėl techninių ribojimų tikrojo ženklo kodo negali teisingai apdoroti kuri nors programa.

kòdų lentẽlė. Koduotė, pateikta lentelės pavidalu.

koduõtė. Ženklų kodavimas, vienareikšmiškai apibrėžiantis tam tikro rinkinio ženklų kodus.

komándų klavìšai. Klavišai arba klavišų kombinacijos, kuriuos paspaudus vykdoma tam tikra komanda, nepriklausomai nuo to, kuris langas tuo metu atvertas arba kur yra žymeklis.

kombinãcinis diakrìtinis žénklas. Diakritinis nulinio pločio ženklas. Kai toks ženklas rodomas ekrane arba spausdinamas, jis uždedamas ant prieš jį esančios raidės ir taip gaunama raidė su diakritiniu ženklu.

kompiuterìnis vertìmas. Automatinis vertimas iš vienos kalbos į kitą naudojant vertimo programą – vertyklę.

kompiuterizúotas vertìmas. Vertimas iš vienos kalbos į kitą, kai vertėjas naudojasi kompiuterinėmis vertimo priemonėmis, pavyzdžiui, vertimo atmintimi, terminų bankais, lokalizavimo programomis.

komplèksinis testãvimas. Visų programos komponentų (dialogo, kompiuterio pranešimų, žinyno ir kitų tekstų) testavimas veikiančioje programoje siekiant patikrinti jų suderintumą ir sąveiką.

kompozìcinė ženklų sekà. Unikodo ženklų seka vienam ženklui suformuoti iš kelių ženklų: pirmasis sekos ženklas yra raidė, o po jos eina vienas arba keli nulinio pločio kombinaciniai diakritiniai ženklai.

kontèkstas. Informacija, esanti frazės išorėje, reikalinga tam, kad ta frazė būtų teisingai suprantama ir verčiama.

kontèkstinis žinýnas. Elektroninis žinynas, į kurį galima patekti iš tuo metu veikiančios programos dialogo langų. Patenkama į žinyno vietą, susijusią su tuo langu.

kultūros elemeñtas. Kompiuteriui skirtas duomenų elementas, priklausomas nuo kalbos, valstybės, teritorijos ir kt. kultūros aplinkybių.

kultūros specifikãcija. Kultūros ypatumų aprašymas, skirtas panaudoti kompiuteryje. Aprašomi kultūros elementai, kuriuos turi atitikti aparatinė ir programinė įranga, skirta konkrečiai kultūrinei terpei: kalbai, valstybei, teritorijai.

l10n [sk: el-dešimt-en]. Programuotojų žargone vartojama žodžio „lokalizavimas“ santrumpa. Šio simbolinio žymens pirmoji ir paskutinė raidės yra angliško žodžio „localization“ pirmoji ir paskutinė raidės, o skaičius 10 rodo, kiek šio žodžio raidžių praleista santrumpoje tarp pirmosios ir paskutinės raidės.

lãpo formãtas. Popieriaus lapo matmenys.

leksemà. Mažiausias prasminis teksto elementas.

leksikogrãfinis rikiãvimas. Žodžių arba eilučių rikiavimas pagal abėcėlę.

lèksikos anãlizė. Leksemų (vardų, bazinių žodžių, skaičių, keliais ženklais užrašytų operacijų ženklų) aptikimas programos, parašytos programavimo kalba, arba scenarijaus tekste ir jų kodavimas, kad toliau apdorojant tekstą jos būtų atpažįstamos tiesiogiai, be analizės.

lèksikos klaidà. Klaida, kai analizuojama leksema neatitinka tokio tipo leksemos sudarymo taisyklių.

lietùvinimas. Programinės įrangos, jos dokumentacijos, saityno išteklių vertimas ir adaptavimas taip, kad jie visiškai tiktų darbui lietuviškoje aplinkoje (Lietuvoje).

Lyg2. Žr. antrojo lygio klavišas.

Lyg3. Žr. trečiojo lygio klavišas.

ligatūrà. Dviejų arba daugiau ženklų derinys, atvaizduotas vienu piešiniu ir šrifte turintis vieną glifą.

lituanizãvimas. Žr. lietuvinimas.

lokãlė. Kompiuteryje ir jo programinėje įrangoje naudojamų elementų, priklausančių nuo kalbos ir kultūros normų, visuma.

lokãlės elemeñtas. Kultūros elementas, kuris yra įtrauktas į formalųjį kultūros elementų aprašą – lokalę.

lokãlės identifikãtorius. Operacinėse sistemose „Windows“ naudojamas 32 bitų lokalės kodas. Lietuvos lokalės identifikatorius yra 00000427 (šešioliktainis).

lokalizãcija. 1. Programinės įrangos adaptavimo kultūrinei ir kalbinei aplinkai proceso rezultatas. **2.** Programos lokalizuota atmaina.

lokalizãcijos klaidà. Programinės įrangos kurios nors savybės neatitikimas lokalei, kai ta savybė yra internacionalizuota.

lokalizãcijos komentãras. Į lokalizuojamųjų išteklių failą įrašytas specialiais ženklais pažymėtas aiškinamasis tekstas, neturinis tiesioginės įtakos lokalizuojamos programos arba svetainės veikimui, skirtas tik lokalizuotojams.

lokalizãcijos testãvimas. Lokalizacijos patikrinimas, ar jos tekstai išversti teisingai ir taisyklingai, ar lokalizacija atitinka kultūrinius reikalavimus.

lokalizãvimas. Programinės įrangos pritaikymas tam tikrai kalbos ir kultūros aplinkai.

lokalizãvimo prográminė į̃ranga. Programinė įranga, naudojama lokalizavimo procesui palengvinti ir iš dalies automatizuoti.

lokalizúojamieji ìštekliai. Programos ištekliai, į kuriuos sudėti duomenys, priklausomi nuo kalbos ir kultūros.

lokalizúojamųjų ìšteklių metainformãcija. Informacija, skirta apibūdinti lokalizuojamųjų išteklių duomenis: jų kontekstą, paskirtį, tipą ir pan.

lokalizúojamųjų ìšteklių pateikìmo formãtas. Lokalizuojamuose ištekliuose laikomų duomenų apipavidalinimo būdas.

lotỹnų abėcẽlė. 1. Abėcėlė, sudaryta iš lotyniškų raidžių, pvz., anglų, latvių, lenkų, lietuvių, ispanų, prancūzų, vokiečių. **2.** Ženklų rinkiniai, turintys jiems skirtas 8 bitų koduotes: lotynų 1-oji, lotynų 2-oji, lotynų 3-oji, ... lotynų 16-oji.

lotỹnų abėcẽlės papìldomas plėtinỹs. Raidės, kurių unikodo kodai yra U+1E00 – U+1EFF. Tai lotynų abėcėlės raidės, nesančios pagrindinėje lotynų abėcėlėje ir nepatekusios į lotynų 1-ąją abėcėlę arba lotynų abėcėlės plėtinį.

lotỹnų abėcẽlės plėtinỹs. Raidės, kurių unikodo kodai yra U+0100 – U+024F. Tai svarbesnės lotynų abėcėlės raidės, nesančios pagrindinėje lotynų abėcėlėje ir nepatekusios į lotynų 1-ąją abėcėlę.

lotỹnų rãšmenys. Lotynų rašto kalbose (pvz., anglų, latvių, lietuvių, vokiečių) vartojami rašmenys.

LST standártas. Lietuvos Respublikos nacionalinis standartas.

mažė́jantys báitai. Unikodu koduoto ženklo baitų išdėstymo būdas, kai pirmiau eina šešioliktukų (šešioliktainės skaičiavimo sistemos) baitas, po jo – vienetų. Žymimas BE, pvz., UTF-16BE. Žr. dar baitų eiliškumo ženklas. Plg. didėjantys baitai.

meniù elemeñtas, meniù pùnktas. Komandos arba meniu grupės pavadinimas.

meniù júosta. Į eilę išdėstytų meniu elementų grupė.

meniù. Operacijų (funkcijų) arba jų grupių, vadinamų meniu elementais arba meniu punktais, pavadinimų sąrašas.

metakalbà. Kalba, skirta kitoms kalboms aprašyti.

MIME protokòlas [sk. mime...] (angl.: MIME = Multi-Purpose Internet Mail Extensions). Elektroninio pašto protokolas, papildantis laiškų siuntimo SMTP protokolą taip, kad elektroniniu laišku būtų galima siųsti įvairių tipų duomenis: 8 bitų tekstą, vaizdo ir garso failus, kitus dvejetainius failus prieš tai jų neperkodavus į 7 bitų ASCII koduotę.

MIME tìpas. Elektroninio laiško arba kurios nors jo dalies duomenų tipas, apibrėžtas ir tos dalies antraštėje aprašytas pagal MIME protokolo taisykles. Yra apibrėžti tokie tipai: teksto, grafikos, garso, vaizdo, kitokių duomenų.

MO fáilas [sk. mo...] (angl. MO = Machine Object). PO failo sukompiliuotas dvejetainis pavidalas, skirtas platinti kartu su programa.

mnemòninis vařdas. Vardas, sudarytas taip, kad jį būtų lengva įsiminti.

nùlinio plõčio žénklas. Ženklas, neturintis pločio. Kai rodomas ekrane arba spausdinamas, jis uždedamas ant prieš jį esančio ženklo. Žr. dar kombinacinis diakritinis ženklas, kompozicinė ženklų seka.

numatýtoji kalbà. Kalba, kurios lokalei suderinta operacinė sistema.

numatýtoji koduõtė. Koduotė, parenkama tekstui rodyti, kai kartu su juo nėra pateiktos informacijos apie jo koduotę.

nusãkomoji kultũros specifikãcija. Neformalizuotas kultūros ypatumų aprašymas, skirtas panaudoti kompiuteryje.

pagrindìnė lotỹnų abėcėlė. Pagrindinę lotynų abėcėlę sudaro 26 didžiosios ir mažosios raidės: Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz.

pakaitõs simbolis. Simbolis, vartojamas reguliariuosiuose reiškiniuose ir žymintis eilutės vietą, į kurią galima įterpti bet kurį vieną ar kelis ženklus.

pamatìnė raĩdė. Raidė, kurią modifikavus gaunama kita raidė. Pavyzdžiui, pridėjus diakritinį ženklą.

parametrizúota eilùtė. Išteklių eilutė, kurioje yra pavartotas parametras – reikšmė, įterpiama į eilutę programą vykdant.

paukščiùkas. Ženklas ˇ. Kodai: 142 (Windows-1257, dešimtainis), U+02C7.

PHP kalbà [sk. pė-ha-pė...]. Scenarijų programavimo kalba, skirta interneto svetainėms kurti.

PO fáilas [sk. po...] (angl. PO = Portable Object). Failas, kuriame laikomi lokalizuotos programos lokalizuojamieji ištekliai. Gaunamas iš PO failo papildžius kiekvieną jo įrašą lokalizuotos eilutės tekstu.

POSIX standártai [sk: pos-iks...] (angl. POSIX = Portable Operating System Interface for Unix). Standartų grupė, apibrėžianti operacinės sistemos sąsajas tarp joje veikiančių programų bei tarnybų.

POT fáilas [sk. pot...] (angl. POT = Portable Object Template). Failas, kuriame laikomi originalios programos lokalizuojamieji ištekliai.

pozìcinis glìfas. Ženklo glifas, priklausantis nuo ženklo vietos žodyje.

prìeigos klavìšai. Alternatyvos ir ženklo klavišų kombinacija, dubliuojanti meniu ir kitų komandų funkcijas.

pseudovertìmas. Originalių programos išteklių eilučių papildymas numeriais ir savitosiomis raidėmis, siekiant nustatyti, kur bus matoma išversta eilutė, ar teisingai vaizduojamos savitosios raidės.

QWERTY klaviatūrà. Klaviatūra, kurios viršutinė raidžių klavišų eilė prasideda raidėmis QWERTY.

QWERTZ klaviatūrà. Klaviatūra, kurios viršutinė raidžių klavišų eilė prasideda raidėmis QWERTZ. Naudojama Čekijoje, Slovakijoje, Vengrijoje, Vokietijoje.

raĩdė su diakrìtiniu žénklu. Raidė, gauta iš kitos raidės, pridėjus prie jos diakritinį ženklą.

raĩdė. Abėcėlės elementas. Kiekviena raidė priklauso kuriam nors raštui (pvz., lotynų, kirilicos) ir gali būti kelių kalbų abėcėlėse.

raĩdę modifikúojantis žénklas. Ženklas, pridedamas prie raidės iš dešinės pusės ir modifikuojantis tos raidės reikšmę.

raĩdžių lỹgio páisymas. Didžiųjų ir jas atitinkančių mažųjų raidžių laikymas skirtingomis.

raĩdžių lỹgis. Raidžių suskirstymas į du lygius: didžiąsias ir mažąsias.

rašýbos tìkrinimo programà. Žr. rašybos tikrintuvė.

rašýbos tikrintùvė. Programa arba rašyklės komponentas, tikrinantis rašybą: ar žodis yra jos turimame žodyne, ar leistina jo forma, kitus nesudėtingus rašybos elementus.

rãšmenys. 1. Tam tikro rašto ženklai. **2.** Parinktis, nurodanti raštą, pvz., lotynų, kirilica, hebrajų.

rašmuõ. Rašto ženklas, vartojamas tam tikrame rašte, pavyzdžiui, raidė, skaitmuo, skyrybos ženklas.

rãštas, rãšto sistemà. Rašmenų rinkinys kartu su rašybos taisyklėmis, reikalingas tam tikros kalbos ar kalbų grupės tekstams užrašyti.

rãšto sistemà. Žr. raštas.

rãšto žénklas. Žr. rašmuo.

RFC (angl. RFC = Request For Comments). Interneto inžinerinių uždavinių sprendimo komiteto (IETF), rengiančio atviruosius standartus, dokumentas.

rikiãvimas. Objektų išdėstymas į eilę pagal kurį nors jų parametrą: skaičių – pagal jų dydį, žodžių – pagal raidžių rikiavimo eilę abėcėlėje ir pan.

rikiãvimo eilẽ. Taisyklės, nusakančios, kokia eile turi būti išdėstytos rikiavimo parametro reikšmės, neturinčios aritmetiškai palyginamų verčių, pavyzdžiui, rašmenys.

saitýno lietùvinimas. Saityno svetainių, protokolų ir paslaugų pritaikymas darbui lietuviškoje aplinkoje (Lietuvoje). Darbas susideda iš dviejų dalių: 1) saityno protokolų, serverių ir klientų programinės įrangos pritaikymas, kad jie teisingai apdorotų lietuviškus rašto ženklus; 2) svetainių vertimas į lietuvių kalbą ir adaptavimas.

semántika. Taisyklės, apibrėžiančios sakinio prasmę ir santykį tarp jo žodžių.

semántikos klaidà. Prasminė klaida programoje, padaryta nenusižengiant sintaksės taisyklėms.

siñtaksė. Taisyklių, apibrėžiančių sakinio struktūrą, visuma.

siñtaksės anãlizė. Programos, parašytos programavimo kalba, arba scenarijaus sintaksės konstrukcijų (aprašų, komandų, sakinių, reiškinių ir kt.) atpažinimas.

siñtaksės klaidà. Klaida, kai tekstas neatitinka tam tekstui taikomų sintaksės taisyklių.

sintètinė kalbà. Kalba, kurioje santykiai tarp žodžių išreiškiami įvairiomis tų pačių žodžių gramatinėmis formomis, žodžių tvarka sakinyje dažniausiai laisva, pvz., lietuvių kalba. Plg. analitinė kalba.

skiemenãvimas. Žodžių skaidymas į skiemenis. Naudojamas kalbos analizėje, kai reikia žodį kelti į kitą eilutę.

skìrti didžiąsias ir mažąsias raidès. Paieškos komandos parinktis, nurodanti, kad, tikrinant ieškomo teksto atitikimą, didžiąsias ir mažąsias raides reikia laikyti skirtingomis. Lokalizuojant programą, turinčią teksto paieškos komandą, reikėtų patikrinti, ar ji tikrai mažąsias lietuvių k. abėcėlės raides su diakritiniais ženklais laiko tapačiomis jas atitinkančioms didžiosioms.

SMTP protokòlas. Elektroninio pašto protokolas, naudojamas elektroniniams laiškams persiųsti tarp kelių interneto serverių.

supàprastintoji kalbà. Minimalus kalbos poaibis, turintis ribotą žodyną ir ribotas raiškos priemones. Rekomenduojama vartoti programinėje įrangoje ir jos dokumentacijoje, kad būtų galima lengviau ją lokalizuoti.

surogãtas. Kurio nors objekto nevisavertis pakaitalas. Pavyzdžiui, koduotėje nesantis ženklas pakeičiamas kitu į jį panašiu ženklu, paveikslas, kurio persiųsti arba pavaizduoti nėra galimybės – supaprastintu paveikslu arba tik paveikslo pavadinimu. Paprastai iš surogato negalima atkurti originalo.

svetaĩnės lokalizãvimas. Svetainės visapusiškas pritaikymas tam tikrai kalbinei ir kultūrinei aplinkai.

šnekõs atpažinìmas. Automatinis šnekos (sakytinės kalbos) atpažinimas ir užrašymas tekstu.

šnekõs sintezãtorius. Programa arba įrenginys (įtaisas), tekstą paverčiantis garsu ir jį skaitantis balsu.

taškaĩ cólyje. Taškų, kuriais pavaizduota vieno colio ilgio linija, skaičius. Žymima: tašk./colyje. Spausdintuvo, spaudinio, ekrano ir jo rodomo vaizdo kokybės matas. Lazeriniai spausdintuvai spausdina 300, 600, 1200 taškų/colyje.

TBX formãtas (angl. TBX = Term Base eXchange). Terminijos duomenų struktūrizavimo ir mainų naudojant XML kalbą formatas. Standartizuotas Lokalizavimo pramonės standartų asociacijos LISA specifikacija TBX ir tarptautiniu standartu ISO 30042.

tèksto kryptìs. Teksto rašymo ir skaitymo kryptis: iš kairės į dešinę, iš dešinės į kairę, iš viršaus į apačią. Iš dešinės į kairę rašoma arabų ir hebrajų kalbomis, iš viršaus į apačią – kinų. Žr. dar dvikryptė kalba.

tèksto segmeñtas. Versti ir lokalizavimo priemonėms apdoroti parengta lokalizuojamųjų išteklių teksto dalis.

tikslì atitiktìs. Išteklių eilutės vertimas, visiškai (100 proc.) atitinkantis originalios eilutės, esančios vertimų atmintyje, vertimą.

tìkslintina eilùtė. Preliminariai išversta programos išteklių eilutė, kurios vertimą dar reikia tikrinti ir tikslinti.

TMX formãtas (angl. TMX = Translation Memory eXchange). Nuo programinės įrangos gamintojo nepriklausomas XML kalba pagrįstas formatas, skirtas vertimo atminties duomenų, sukurtų kompiuterizuoto vertimo ir lokalizavimo priemonių, mainams. Standartizuotas Lokalizavimo pramonės standartų asociacijos LISA specifikacija TMX.

transkrìpcija. Vienos kalbos raidžių (žodžių) perrašymas kitos kalbos rašmenimis pagal tarimą.

transliterãcija. Vienos kalbos raidžių perrašymas kitos kalbos raidėmis taip, kad būtų galimas vienareikšmis atvirkštinis perrašymas.

trẽčiojo lỹgio klavìšas. Klavišas, kurį laikant paspaustą renkami trečiojo lygio (registro) ženklai. Lietuviškoje standartinėje klaviatūroje žymimas Lyg3.

trùpmenos skirtùkas. Ženklas, skiriantis dešimtainės trupmenos trupmeninę dalį nuo sveikosios. Lietuvoje – kablelis.

tū́kstančių skirtùkas. Skirtukas didelio (ilgo) skaičiaus skaitmenis grupuoti po tris. Priklauso nuo kalbos ir lokalės. Gali būti taškas, kablelis, tarpas, jungiamasis tarpas. Lietuvių kalboje tūkstančių skirtukas yra taškas arba tarpas.

ùmliautas. Žr. du taškai viršuje.

unikòdas. Ženklų koduotė, kurioje vienam ženklui skiriama 16 bitų (du baitai – oktetai).

unikòdo privačiójі sritìs. Laisva unikodo kodų sritis, kuriai nėra priskirtų ženklų.

unikòdo surogãtų sritìs. Kodų sritis U+D800 – U+DFFF unikode, skirta 32 bitų kodus turintiems 32 bitų kodus iš intervalo U+10000 – U+10FFFF, koduoti. Vienas ženklas koduojamas dviem surogatų srities kodais.

unikòdo ženklų̃ normalizãvimas. Unikodo ženklų vienareikšmis išreiškimas kodų seka.

UTF-8 kodãvimas [sk: u-tė-ef-aštuoni...]. Unikodo (ISO/IEC 10646) ženklų vaizdavimas vienu arba kelias 8 bitų kodais.

UTF-16 kodãvimas. Unikodo (ISO/IEC 10646) ženklų vaizdavimas 16 bitų kodais.

UTF-32 kodãvimas. Unikodo (ISO/IEC 10646) ženklų vaizdavimas 32 bitų kodais.

ùžrašas ant klavìšo. Ant klaviatūros klavišo užrašytas juo renkamas ženklas, klavišo pavadinimas ir (arba) piktograma.

vaĺdantysis žõdis. Žodis, nuo kurio priklauso kitų frazės žodžių gramatinės formos.

vaĺdymo klavìšas. Klaviatūros klavišas, kuris, paspaustas kartu su kitu valdymo klavišu, pakeičia to klavišo funkciją, suteikdamas jam valdymo (komandos) funkciją. Lietuviškoje standartinėje klaviatūroje žymimas Vald.

vaĺdymo kòdai. Ženklų koduotėse esantys kodai, skirti valdymo funkcijoms atlikti.

vaĺdymo žénklas. Valdymo veiksmą nurodantis ženklas, esantis ženklų koduotėje.

valiùtos formãtas. Pinigų sumos užrašymo pavidalas, nurodantis, kaip grupuojami didelių skaičių skaitmenys (dažniausiai po tris) ir kokiu ženklu skiriamos jų grupės, kokiu ženklu skiriamos piniginio vieneto šimtosios dalys (pvz., centai), koks valiutos ženklas arba santrumpa ir kur ji rašoma: prieš skaičių ar po jo.

valiùtų ženklaĩ. Specialieji ženklai, skirti valiutoms žymėti. Lietuvoje standartizuotose 8 bitų koduotėse yra šie ženklai: € euras, $ JAV doleris, ¢ centas, £ svaras sterlingų, ¤ neįvardytos valiutos ženklas.

valstýbės kòdas. Dviem raidėmis pažymėtas valstybės pavadinimas, dažniausiai sudarytas iš tos valstybės pavadinimo raidžių.

vertìmo atmintìs. Duomenų bazė, kurioje laikomi teksto segmentai ir jų vertimai tam, kad vertimus būtų galima panaudoti iš naujo.

vertìmo klaidà. Lokalizacijos klaida, kai išversta frazė neatitinka prasmės programoje.

vertìmo programà. Žr. vertyklė.

vertìmo víenetas. Vertimo atmintyje įrašyta susieta originalo ir vertimo segmentų pora.

vertỹklė. Programa, verčianti tekstą iš vienos kalbos į kitą (pvz., iš vokiečių į lietuvių).

vienbaĩtė koduõtė. Koduotė, kurioje vienas ženklas koduojamas vienu baitu.

vienkaĨbis rikiãvimas. Daugiakalbio teksto rikiavimas pagal vienos kalbos rikiavimo taisykles. Rezultatas gali skirtis nuo daugiakalbio rikiavimo.

vienkrỹptė kalbà. Kalba, kurioje rašoma viena kryptimi.

XLIFF fáilas [sk: iks-lif failas] (angl. XLIFF = XML Localization Interchange File Format). XML pagrindu sukurtas failų formatas lokalizavimo duomenų mainams tarp programinės įrangos kūrėjų ir lokalizuotojų arba tarp įvairių lokalizavimo automatizavimo programinių priemonių.

XML kalbà (angl. XML = Extensible Markup Language). Universali dokumentų ženklinimo kalba, skirta dokumentų struktūrai aprašyti.

žénklo kòdas. Ženklą identifikuojantis natūralusis skaičius (kodas) tam tikroje koduotėje.

žõdis. 1. Svarbiausias reikšminis kalbos vienetas. **2.** Mažiausias adresuojamas kompiuterio atminties vienetas.

žõdžių kėlĨimas. Žodžio dalies, netilpusios į eilutę, kėlimas į kitą eilutę.

2 priedas. Ženklų ir klavišų pavadinimai

41 lentelė. Lietuvių kalbos abėcėlės raidžių kodai ir pavadinimai

Raidė	Kodas unikode		Angliškas pavadinimas pagal LST ISO/IEC 10646-1	Lietuviškas pavadinimas pagal LST ISO/IEC 8859-13	Pavadinimas šnekoje
	Didž. (A)	Maž. (a)			
A	0041	0061	LATIN CAPITAL LETTER A	Didžioji raidė A	a
Ą	0104	0105	LATIN CAPITAL LETTER A WITH OGONEK	Didžioji raidė Ą	a nosinė
B	0042	0062	LATIN CAPITAL LETTER B	Didžioji raidė B	bė
C	0043	0063	LATIN CAPITAL LETTER C	Didžioji raidė C	cė
Č	010C	010D	LATIN CAPITAL LETTER C WITH CARON	Didžioji raidė Č	čė
D	0044	0064	LATIN CAPITAL LETTER D	Didžioji raidė D	dė
E	0045	0065	LATIN CAPITAL LETTER E	Didžioji raidė E	e
Ę	0118	0119	LATIN CAPITAL LETTER E WITH OGONEK	Didžioji raidė Ę	e nosinė
Ė	0116	0117	LATIN CAPITAL LETTER E WITH DOT ABOVE	Didžioji raidė Ė	ė
F	0046	0066	LATIN CAPITAL LETTER F	Didžioji raidė F	ef
G	0047	0067	LATIN CAPITAL LETTER G	Didžioji raidė G	gė
H	0048	0068	LATIN CAPITAL LETTER H	Didžioji raidė H	ha
I	0049	0069	LATIN CAPITAL LETTER I	Didžioji raidė I	i
Į	012E	012F	LATIN CAPITAL LETTER I WITH OGONEK	Didžioji raidė Į	i nosinė
Y	0059	0079	LATIN CAPITAL LETTER Y	Didžioji raidė Y	i ilgoji
J	004A	006A	LATIN CAPITAL LETTER J	Didžioji raidė J	jot
K	004B	006B	LATIN CAPITAL LETTER K	Didžioji raidė K	ka
L	004C	006C	LATIN CAPITAL LETTER L	Didžioji raidė L	el
M	004D	006D	LATIN CAPITAL LETTER M	Didžioji raidė M	em
N	004E	006E	LATIN CAPITAL LETTER N	Didžioji raidė N	en
O	004F	006F	LATIN CAPITAL LETTER O	Didžioji raidė O	o
P	0050	0070	LATIN CAPITAL LETTER P	Didžioji raidė P	pė
R	0052	0072	LATIN CAPITAL LETTER R	Didžioji raidė R	er
S	0053	0073	LATIN CAPITAL LETTER S	Didžioji raidė S	es
Š	0160	0161	LATIN CAPITAL LETTER S WITH CARON	Didžioji raidė Š	eš

Raidė	Kodas unikode		Angliškas pavadinimas pagal LST ISO/IEC 10646-1	Lietuviškas pavadinimas pagal LST ISO/IEC 8859-13	Pavadinimas šnekoje
	Didž. (A)	Maž. (a)			
T	0054	0074	LATIN CAPITAL LETTER T	Didžioji raidė T	tė
U	0055	0075	LATIN CAPITAL LETTER U	Didžioji raidė U	u
Ų	0172	0173	LATIN CAPITAL LETTER U WITH OGONEK	Didžioji raidė Ų	u nosinė
Ū	016A	016B	LATIN CAPITAL LETTER U WITH MACRON	Didžioji raidė Ū	u ilgoji
V	0056	0076	LATIN CAPITAL LETTER V	Didžioji raidė V	vė
Z	005A	007A	LATIN CAPITAL LETTER Z	Didžioji raidė Z	zė
Ž	017D	017E	LATIN CAPITAL LETTER Z WITH CARON	Didžioji raidė Ž	žė

Mažųjų raidžių žodinius pavadinimus galima gauti iš didžiųjų raidžių pavadinimų pakeičiant žodį *Didžioji* į *Mažoji* (CAPITAL į SMALL angliškame pavadinime).

Šnekoje mažosios ir didžiosios raidės vadinamos vienodai. Dėl to santrumpose vengiama mažųjų raidžių. Kada reikia nurodyti, jog tai yra mažoji arba didžioji raidė, tai pasakomas ir šis pažymimasis žodis.

Tariant santrumpas pažymimieji žodžiai *ilgoji* ir *mažoji* praleidžiami. Laikoma, kad jie suprantami iš konteksto. Kada reikia pasakyti, jog tai ilgoji arba nosinė, tai pasakomas ir šis pažymimasis žodis.

Raidės *x*, *y* ir *z* matematikoje turi ypatingą paskirtį kintamiesiems ir koordinačių ašims žymėti. Čia jos vadinamos taip: *iksas*, *igrekas*, *zet*.

42 lentelė. Valdymo klavišų pavadinimai

Užrašas ant klavišo	Klavišo vieta	Klavišo pavadinimas tekste	
		Lietuvių kalba	Anglų kalba
Gr		Gr	Esc
F1, F2, ..., Fn		F1, F2, ..., Fn	F1, F2, ..., Fn
⌫		Naikinti	Backspace
Tab		Tab	Tab
Didž		Didž	CapsLock
Lyg2		Lyg2	Shift
Lyg2	Dešinėje	Lyg2 (dešinysis)	Right Shift
Lyg3		Lyg3	Right Alt
Vald		Vald	Ctrl
Vald	Dešinėje	Vald (dešinysis)	Right Ctrl

Užrašas ant klavišo	Klavišo vieta	Klavišo pavadinimas tekste	
		Lietuvių kalba	Anglų kalba
Alt		Alt	Alt
Įvesti		Įvesti	Enter
	Kairėje	Windows (kairysis)	Left Windows
	Dešinėje	Windows (dešinysis)	Right Windows
		Tarpas	Space
		Programa	Application
		Pagalba	Help
Sp	Redagavimo srityje	Sp	Prnt Scrn
Slinkti	Redagavimo srityje	Slinkti	Scroll Lock
Pauzė	Redagavimo srityje	Pauzė	Pause
		Sist.	Sys Req
Įterpti	Redagavimo srityje	Įterpti	Insert
Prad	Redagavimo srityje	Prad	Home
Psl↑	Redagavimo srityje	Psl↑	Page Up
Šal	Redagavimo srityje	Šal	Delete
Pab	Redagavimo srityje	Pab	End
Psl↓	Redagavimo srityje	Psl↓	Page Down
↑	Redagavimo srityje	Aukštyn	Up
↓	Redagavimo srityje	Žemyn	Down
←	Redagavimo srityje	Kairėn	Left
→	Redagavimo srityje	Dešinėn	Right
Skaitm	Skaitmenų srityje	Skaitm	Num Lock
/	Skaitmenų srityje	/ (sk.)	Num /
*	Skaitmenų srityje	* (sk.)	Num *
–	Skaitmenų srityje	– (sk.)	Num –
+	Skaitmenų srityje	+ (sk.)	Num +
Įvesti	Skaitmenų srityje	Įvesti (sk.)	Num Enter
Šal	Skaitmenų srityje	Šalinti (sk.)	Num Del
0, 1, ..., 9	Skaitmenų srityje	0 (sk.), 1 (sk.), ..., 9 (sk.)	Num 0, Num 1, ..., Num 9

Pastabos:

1. Tuščias klavišo vietos stulpelis reiškia, kad klavišas yra pagrindinėje klaviatūros srityje ir jo tikslesnės vietos nėra būtina nurodyti.
2. Klaviatūroje gali nebūti kai kurių klavišų (pvz., *Pagalba*).
3. Redagavimo klavišų dublikatai (*Prad, Pab* ir kt.), esantys skaitmeninėje klaviatūros srityje, atlieka tas pačias funkcijas, kaip ir jų originalai, todėl tekstuose jie įvardijami vienodai.

3 priedas. Ženklų rinkimas kodais

Aštuonbitės koduotės kodais. „Windows“ sistemose visus lokalės koduotėje (lietuvių kalbos atveju – „Windows-1257“) esančius ženklus galima rinkti kodais. Esant skaitmeninei klaviatūrai būsenai, laikant paspaustą alternatyvos klavišą skaitmeninėje klaviatūros dalyje renkamas dešimtainis ženklo kodas, priekiniais nuliais papildytas iki keturių skaitmenų. Atleidus alternatyvos klavišą, ekrane atsiranda surinktą kodą atitinkantis ženklas. Pavyzdžiui, raidė Ä gaunama surinkus 0196, nes 196 yra jos dešimtainis kodas.

Dešimtainiai kodai paprastai būna užrašyti kodų lentelės langeliuose (žr. 18 pav.).

Unikodo kodais. Operacinėse sistemose „Windows XP“, „Windows Vista“, „Windows 7“ yra galimybė ženklus rinkti unikodo kodais. Sistemoje turi būti įjungta šešioliktainių ženklų kodų įvedimo funkcija (registro lauke *HKEY_Current_User\Control Panel\Input Method* nustatyta *"EnableHexNumpad"="1")*.

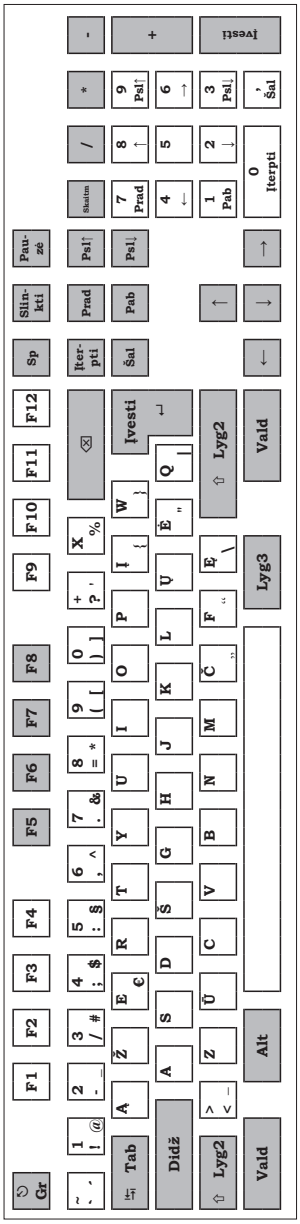
Laikant paspaustą alternatyvos klavišą skaitmeninėje klaviatūros dalyje reikia surinkti pliuso ženklą, po to – norimo ženklo šešioliktainį kodą. Atleidus alternatyvos klavišą ženklas pasirodys ekrane. Didžiosios ir mažosios raidės šešioliktainiame kode lygiavertės, skaitmenis galima rinkti bet kurioje klaviatūros dalyje.

Taip galima rinkti ženklus bet kuriame programos, kuri operuoja unikodu, lange: skaičiuoklės, pateikčių rengyklės, pašto programos, naršyklės, tekstų rengyklės, išskyrus „Word“ programą.

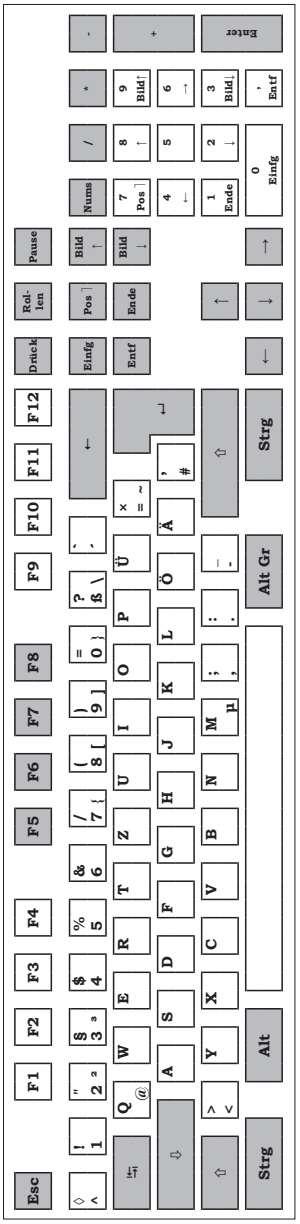
Ženklas teisingai surenkamas ir tada, kai ženklo piešinio nėra rinkimo metu naudojamame šrifte. Paprastai tokiu atveju vietoj ženklo rodomas kvadratėlis. Tikras ženklas gali atsirasti pakeitus šriftą arba kvadratėlį nukopijavus ir perkėlus į kitą programą, turinčią tinkamą šriftą.

Tekstų rengyklėje „Word“ – sava tvarka. Norint rengiamame dokumente gauti norimą ženklą, reikia surinkti ženklo kodą (be pliuso priekyje), pavyzdžiui, 03B1, po to, esant žymekliui po paskutinio kodo ženklo, paspausti Alt+X. Kodas virs ženklu. Vėl paspaudus Alt+X, ženklas virs kodu. Taip galima ne tik surinkti ženklą kodu, bet ir pamatyti bet kurio dokumente esančio ženklo kodą.

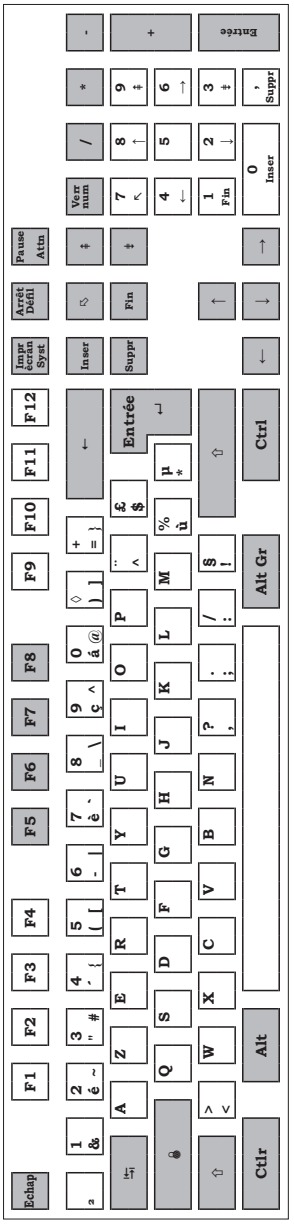
4 priedas. Klaviatūros



59 pav. Lietuvių kalbos klaviatūra (LST 1582)



60 pav. Vokiečių kalbos klaviatūra



61 pav. Prancūzų kalbos klaviatūra

5 priedas. Dažniau pasitaikančios programų internacionalizavimo klaidos

Siekiant naujų programų testuotojams suteikti galimybę rasti daugiau internacionalizavimo klaidų ir paspartinti jų paiešką, buvo parengtas klausimynas. Tai užuominos, kur ir ką reikia tikrinti.

Klaidos, suskirstytos į tris grupes, pateiktos tolesniuose skyreliuose.

1. Rašto ženklai

Visavertis visų lietuviškų raidžių vartojimas visuose kompiuteriu ruošiamuose ar tvarkomuose dokumentuose ir kompiuterio ekrane matomuose tekstuose yra esminis dalykas, todėl jį reikia išsamiai patikrinti.

Lietuvos standartais įteisintos dvi aštuonių bitų koduotės: ISO 8859-13 ir „Windows-1257“. Nors jos skirtingos, visų lietuvių kalbos abėcėlės raidžių kodai nesiskiria, todėl problemų jas vartojant nėra.

Lietuvių kalboje vartojamos kabutės („“) taip pat yra abiejose koduotėse, tačiau jų kodai skiriasi. Todėl ten, kur kartu su tekstu nurodoma ir jo koduotė (pvz., elektroniniuose laiškuose) turi būti numatytas automatinis perkodavimas.

ISO 8859-13 koduotėje nėra brūkšnio. Tekstą perkoduojant į šią koduotę brūkšnys paprastai keičiamas vienu brūkšneliu arba dviem brūkšneliais.

Mobiliuosiuose telefonuose tekstinės žinutės koduojamos unikodu, tačiau čia ženklų aibė ribojama. ETSI ES 202 130 standartas nustato, kad lietuviškose žinutėse privalomos visos 32 lietuvių kalbos abėcėlės raidės ir rekomenduojamos dar trys anglų kalbos abėcėlės raidės *q, w* ir *x*. Lietuviškos kabutės ir brūkšnys į privalomų ženklų aibę neįtraukti.

Klaidos, susijusios su savitųjų lietuviškų raidžių (*ąčęėįšųūž*) vartojimu dokumentuose (rašiniuose, tinklalapiuose, elektroniniuose laiškuose, pokalbių programų žinutėse) greitai pastebimos ir kompiuterių programose jų retai pasitaiko, bet rečiau matomuose programų languose jos dar dažnos. Klausimyne joms skirta daugiau dėmesio.

Visų lietuvių kalbos abėcėlės raidžių vartojimą visur, išskyrus abonento vardą, rašomą elektroninio pašto adrese prieš ženklą @, reglamentuoja Lietuvos ir tarptautiniai standartai. Jiems realizuoti nėra techninių kliūčių, todėl bet koks nesklandumas laikytinas internacionalizavimo klaida.

Toliau pateiksime klausimus testuotojams, susijusius su rašto ženklais, taikytinus programoms, turinčioms klausimuose minimas funkcijas.

1. Ar galima surinkti visas lietuvių kalbos abėcėlės raides, kabutes, brūkšnį, ar visi šie ženklai teisingai rodomi kompiuteriu rengiamame, skaitomame arba modifikuojamame rašinyje, tinklalapyje, tinklalapio pavadinime, elektroniniame laiške, žinutėje ir bet kuriame kitame dokumente ar jo dalyje?
2. Ar galima surinkti visas lietuvių kalbos abėcėlės raides, kabutes ir brūkšnį tinklalapio laukuose (pvz., komentare, anketoje), kuriuos pildo tinklalapio lankytojas, ar juose surinktas tekstas teisingai išsiunčiamas į tinklalapį?
3. Ar galima surinkti visas lietuvių kalbos abėcėlės raides, kabutes, brūkšnį, ar šie ženklai teisingai rodomi tekstui skirtuose dialogo langų laukuose?
4. Ar teisingai persiunčiami laiškai, žinutės ir kiti tekstiniai pranešimai tarp pašto programų, mobiliųjų telefonų ir kitokių įrenginių?
5. Ar teisingai pavaizduojamos visos lietuvių kalbos abėcėlės raidės ir kabutės dokumente, konvertuotame iš vieno formato į kitą? (Pavyzdžiui, tarp DOC, HTML, TXT, XLS formatų.)
6. Ar teisingai keičiamos mažosios raidės didžiosiomis (ir atvirkščiai)?
7. Ar tekstai rikiuojami pagal lietuvių kalbos abėcėlę: *aąbcčdeęėfghiįyjklmnoprsštuųūvz*, ar raidės *aą, eę, ėį, yį, uų, ū* rikiuojant laikomos lygiavertėmis (bet ne *cč, sš, zž*)?
8. Ar paieškose netapatinamos raidės *ą* su *a*, *č* su *c* ir pan.?
9. Ar gali būti vartojamos visos lietuvių kalbos abėcėlės raidės registracijos varduose (išskyrus abonento vardą elektroninio pašto adrese), asmenvardžiuose ir kituose registracijos duomenyse?
10. Ar slaptažodžiuose gali būti vartojamos visos lietuvių kalbos abėcėlės raidės?
11. Ar visos lietuvių kalbos abėcėlės raidės vartojamos kontroliniuose koduose ir iš jų sudaromi lietuvių kalbai būdingi raidžių deriniai?
12. Ar galima atverti tinklalapius, kurių sričių varduose yra raidžių su diakritiniais ženklais (pvz., *www.žalgiris.lt, www.kodėlčius.lt*), ar teisingai rodomi jų adresai?
13. Ar visos lietuvių kalbos abėcėlės raidės gali būti vartojamos failų, aplankų varduose?
14. Ar galima į programą įkelti dokumentą, kurio failo varde vartojamos bet kurios lietuvių kalbos abėcėlės raidės?
15. Ar galima į programą įkelti dokumentą iš aplanko arba įrašyti į aplanką, kurio varde vartojamos bet kurios lietuvių kalbos abėcėlės raidės?

16. Ar pakavimo programa gali teisingai supakuoti ir išpakuoti aplankus ir failus, kurių varduose vartojamos bet kurios lietuvių kalbos abėcėlės raidės?
17. Ar programa (pokalbių, pašto ir kt.) gali persiųsti failą, kurio varde vartojamos bet kurios lietuvių kalbos abėcėlės raidės?
18. Ar galima atverti tinklalapius, kurių aplankų ir failų varduose yra raidžių su diakritiniais ženklais, ar teisingai rodomi jų adresai?
19. Ar programa gali būti įdiegta į aplanką, kurio varde yra savitųjų lietuviškų raidžių?
20. Ar visuose elektroninio laiško laukuose (*Kas, Kam, Tema* ir kt.), išskyrus elektroninio pašto adresą, galima vartoti bet kurias lietuvių kalbos abėcėlės raides?
21. Ar elektroninio laiško tekstas koduojamas Lietuvos standartus atitinkančia koduote ISO 8859-13, „Windows-1257“, ISO 10646 (unikodu) ir jos pavadinimas (iso-8859-13, Windows-1257, UTF-8) įrašomas į elektroninio laiško antraštę?
22. Ar atsakymas į laišką koduojamas ta pačia koduote, kaip ir gautasis laiškas?
23. Ar galima atverti ir teisingai rodyti bei spausdinti laišką, parašytą bet kuria Lietuvos standartus atitinkančia koduote, nurodyta laiško antraštėje?
24. Ar tekstai, įrašomi hiperteksto (HTML) formatu, koduojami Lietuvos standartus atitinkančia koduote ISO 8859-13, „Windows-1257“, ISO 10646 (unikodu) ir jos pavadinimas (iso-8859-13, Windows-1257, UTF-8, UTF-16) įrašomas į tinklalapio antraštę?
25. Ar galima atverti ir teisingai rodyti bei spausdinti tinklalapį, parašytą bet kuria Lietuvos standartus atitinkančia koduote?
26. Ar raidės *Č, č, Š, š, Ž, ž* įrašomos į ISO 8859-13 arba „Windows-1257“ koduote koduojamą hiperteksto (HTML) dokumentą, koduojamos 8 bitų kodais, o ne skaitiniais kodais (*&xxxx*; pavidalu)?
27. Ar išsiunčiamų laiškų antraščių laukai koduojami B arba Q kodu?
28. Ar lokalizuojant programą galima keisti komandų klavišų raides?
29. Ar prieigos klavišai neblokuoja kurio nors trečiojo lygio ženklo, renkamo klaviatūra (t. y. ar galima surinkti visus trečiojo lygio ženklus)?
30. Ar komandų klavišams nenaudojami kitų ženklų klavišai, išskyrus raidžių ir skaitmenų klavišus (pvz., Vald+%)?

Bandytose programose visos su ženklais susijusios klaidos buvo internacionalizavimo klaidos. Reikia tikėtis, kad šiai grupei priklausys ir su ženklais susijusios klaidos kitose programose. Tokių klaidų radus galima jas laikyti internacionalizavimo klaidomis.

Jeigu tikrinamas duomenų persiuntimas telekomunikacijos priemonėmis tarp skirtingų kompiuterių arba tarp programų kompiuterio viduje, tai viename punkte – siuntimo arba gavimo, turi būti etaloninė programa (arba įrenginys), kuri tą funkciją atlieka teisingai, o kitame – testuojamoji. Antraip, aptikus klaidą, bus neaišku, kuri programa klysta.

2. Formatai

Dažniausiai vartojami formatai priskiriami pagrindiniams lokalės elementams. Tai – skaičių, datų, laiko, matavimo vienetų formatai. Jie turėtų būti teisingi ir originalioje programoje, nustatomi pagal parinktą kalbą – dažniausiai paimant jų reikšmes iš operacinėje sistemoje esančių lokalės duomenų.

Rečiau vartojami formatai (pvz., telefono numerio formatas), kurių nebūna lokalėse, turėtų būti lokalizuojamuose ištekliuose. Jeigu jie ne tokie, kokie turi būti lietuviškoje aplinkoje, dar nereiškia, kad tai internacionalizacijos klaida. Reikia patikrinti – gal jie yra lokalizuojamuose ištekliuose arba juos galima parinkti programos nuostatose.

Visus klausimus formulavome taip, lyg būtų testuotajama lokalizuota programa. Tad gavus neigiamą rezultatą dar reikia patikrinti programos nuostatas ir lokalizuojamuosius išteklius.

1. Ar skaičiai į dokumentą rašomi ir rodomi teisingu formatu: trupmenos skirtukas yra kablelis, skaitmenų grupių (tūkstančių) skirtukas – taškas arba tarpas?
2. Ar dialogo langų laukai priima teisingu formatu užrašytus skaičius, rodo ir patys juos generuoja teisingu formatu: trupmenos skirtukas yra kablelis, skaitmenų grupių (tūkstančių) skirtukas – taškas arba tarpas?
3. Ar į dokumentą automatiškai įterpiama data teisingu formatu (trumpuoju ir (arba) ilguoju)?
4. Ar dialogo langų laukai priima teisingu formatu užrašytas datas, rodo ir patys jas generuoja teisingu formatu (trumpuoju ir (arba) ilguoju)?
5. Ar į dokumentą automatiškai įterpiamas laikas teisingu 24 val. formatu?
6. Ar pirmąja savaitės diena laikomas pirmadienis?
7. Ar dokumentų lapų formatai atitinka Lietuvoje galiojančias raštvedybos normas (A3, A4, A5 ir pan.)?
8. Ar puslapio, paraščių, įtraukų dydžiai ir kiti matmenys pateikiami metrinės matų sistemos vienetais (centimetrais, milimetrais)?
9. Ar numatytasis pastraipos pirmosios eilutės įtraukos dydis yra 0,7–0,8 cm?

10. Ar pašto adresų formos atitinka Lietuvos administracinį suskirstymą?
11. Ar pašto adresų komponentai atitinka Lietuvos administracinį suskirstymą?

3. Tekstai

Visi tekstai, išskyrus panaudotus formatuose (6.2 skyr.), originalioje programoje būna originalo kalba. Daugiausia problemų sukelia tekstai, formuojami sujungiant kelias išteklių eilutes.

Eilutės sujungiamos taip, kad jų eilės tvarka ir gramatinės žodžių formos atitiktų originalo kalbos, kuri dažniausiai būna anglų, taisykles. Kitų kalbų taisyklės gali skirtis (taip dažniausiai ir būna). Pavyzdžiui, užrašas prie mygtuko, atšaukiančio paskiausią įterpimą, angliškoje programos versijoje būna „Undo Insert“. Lietuviškai derėtų versti „Atšaukti įterpimą“, bet ištekliuose yra tik atskiros eilutės „Undo“ ir „Insert“, kurios naudojamos ir kitose programos vietose atskirų komandų atskiriems mygtukams pažymėti. Jeigu išversime „Atšaukti“ ir „Įterpti“, tai atskiriems mygtukams tiks, bet vietoj „Undo Insert“ pasirodys užrašas „Atšaukti Įterpti“. O jei „Insert“ išversime kaip „įterpimą“, tai ir ant atskiro įterpimo komandos mygtuko (ir galbūt ne vieno) pasirodys užrašas „įterpimą“. Todėl jeigu po pseudovertimo ekrane pastebima suklijuota eilutė (t. y. turinti du numerius, pvz., „25-Undo 123-Insert“), tai internacionalizavimo klaidos nebus tik tada, jeigu niekur kitur ekrane nebus eilučių, pažymėtų numeriu 123 (šiuo atveju eilutė Nr. 25 gali būti).

Panašių internacionalizavimo klaidų taisymas programos originale nėra sudėtingas – tereikia į lokalizuojamuosius išteklius įtraukti atskiras eilutes su reikalingomis žodžių gramatinėmis formomis. Toks taisymas gali būti labai naudingas: ištaisius klaidas lokalizuotoje suklijuotoje eilutėje ne tik neliks akivaizdžios kalbos klaidos, bet lietuviui lietuviškas užrašas bus suprantamesnis, negu anglui angliškas.

Esant sudėtingiems atvejams naudinga išversti keletą įtartinų vietų, ir, paleidus programą, pažiūrėti, kaip tie vertimai atrodo kompiuterio ekrane.

Apskritai, lokalizavimo pradžioje naudinga išbandyti ir kitokias įtarimus sukeliančias išteklių eilutes (pvz., daugiareikšmius angliškus terminus).

Pateikiame klausimyno fragmentą.

1. Ar visi ekrane matomi tekstai įtraukti į lokalizuojamuosius išteklius?
2. Ar visi išsiunčiami tekstai (pvz., kalendoriaus programos laiškas apie būsimą įvykį) įtraukti į lokalizuojamuosius išteklius?
3. Ar puslapių, lentelių, paveikslų numeriai rašomi pradžioje (prieš santrumpą), pavyzdžiui, *1 pav.*?

4. Ar ekrane matomi elektroninio laiško antraštėje esančių laukų pavadinimai (*Subject, Re, Fwd* ir kt.) įtraukti į lokalizuojamuosius išteklius?
5. Ar pranešimai apie išsiųstą laišką ir gautą laišką įtraukti į lokalizuojamuosius išteklius?
6. Ar atsakymuose ir persiunčiamuose laiškuose pateikiamos informacijos tekstai įtraukti į lokalizuojamuosius išteklius?
7. Ar pranešimai apie nerastą tinklalapį įtraukti į lokalizuojamuosius išteklius?
8. Ar laukuose, sąrašuose ir kitose vietose asmens vardas eina prieš pavardę?
9. Ar yra ekrane matomų registracijos vardų ir asmenvardžių linksniavimo galimybė (pvz., *Sveiki, Jonai; Laiškas persiųstas Jonui*)?
10. Ar yra ekrane matomų būdvardžių ir dalyvių giminės suderinimo su asmens gimine galimybė?
11. Ar žinyne ir kitur pateikiamos iliustracijos, turinčios verstinų tekstų, yra įtrauktos į lokalizuojamuosius išteklius?
12. Ar daiktavardžių linksniai derinami su skaičiais ir turi tris gramatines formas (t. y. ar lokalizacijoje galima padaryti, kad būtų rodoma, pvz., *1 daiktas, 2 daiktai, 10 daiktų*)?
13. Ar tinklalapiai, į kuriuos eina saitai iš dialogo langų, turi lietuviškus variantus? Jei neturi, tai jie turėtų būti lokalizuojami kartu su programa.
14. Ar tinklalapiai, turintys informacijos, būtinai reikalingos programos naudotojui, į kuriuos eina saitai iš programos žinyno, turi lietuviškus variantus? Jei neturi, tai jie turėtų būti lokalizuojami kartu su programa.
15. Ar užrašai telpa į jiems skirtas fiksuoto ilgio vietas?
16. Ar teisinga matavimo vienetų skyryba: tarp skaičiaus ir vieneto pavadinimo ar jo trumpinio yra tarpas (pvz., *8 bitai*, *5 cm*, bet ne *8bitai, 5cm*)?
17. Ar dimensijose vienetai, kurie neįtraukti į lokalizuojamuosius išteklius, skiriami dešininiu brūkšniu (pvz., *b/s*, ne *bps*)?
18. Ar nėra nuostatų, kurių buvimas lokalizuotoje programoje klaidintų jos naudotoją (pvz., siūlymas savaitės dienų pavadinimus rašyti iš didžiųjų raidžių)?
19. Ar tekstų rengyklėse vartojami dokumentų (tarnybinių raštų, laiškų, pažymų, įgaliojimų ir kt.) šablonai įtraukti į lokalizuojamuosius išteklius?

Šis, paskutinis, klausimas labai bendras. Dokumentų šablonų yra daug ir jie sudaro didelę tekstų rengyklių lokalizuojamųjų išteklių dalį, dedamą į atskirus failus. Jų lokalizavimas, tiksliau dokumentų šablonų parengimas, yra specifinis darbas, kurį reikėtų nagrinėti atskirai.

Literatūra

Adomavičiūtė J. (2009) Automobilių interneto svetainių lokalizavimo Lietuvos vartotojams analizė. Kompiuterininkų dienos – 2009, p. 51–59.

Aleknavičienė O., Grumadienė L., Gurskas A., Skirmantas P., Strockis M., Tumasonis V. (2005) Lituanistinis šriftas Palemonas. Tautinių bendrijų namai, Vilnius.

Arenas A. G. (2008) Productivity and quality in the post-editing of outputs from translation memories and machine translation. Localisation Focus. The International Journal of Localization. Vol. 7, Issue 1, p. 11–21.

Allen C. G. (1975) A Manual of European Languages for Librarians. London and New York: Bowker.

Auer S., Dick E. (2007) When does a difference make a difference? A snapshot on global icon comprehensibility. In: Jacko J.A. (Ed.) Human-computer interaction, Pt 2, proc. of 12th International Conference on Human-Computer Interaction. Lecture Notes in Computer Science, 4551, p. 3–12.

Ågerfalk P. J., Fitzgerald B., Olsson H. H., Conchúir O. (2008) Benefits of Global Software Development: The Known and Unknown. In: Wang Q., Pfahl D., Raffo D. (Eds.) Making Globally Distributed Software Development a Success Story, proc. of International Conference on Software Process' 2008, Leipzig. Lecture Notes in Computer Science, 5007, p. 1–9.

Alexin Z., Gyimóthy T., Horváth T., Fábricz K. (1990) Attribute grammar specification for a natural language understanding interface. Attribute Grammars and their Applications, LNCS, vol. 461, p. 313–326.

Apple Computer (1992) Guide to Macintosh Software Localization. Addison Wesley, Reading, Mass.

Atkin S. E. (2001) A Framework for Multilingual Information Processing. Doctoral dissertation. Florida Institute of Technology, Melbourne, Florida.

Baack D., Singh N (2007) Culture and web communications. Journal of Business Research, No. 60, pp. 181–188.

Bargary K. (2006) Translation Web Services – an Implementation for the IGNITE Project. Localisation focus. The international journal for Localisation. Vol. 5, Issue 1, p. 22–23.

Batra S., Bishu R. R. (2007) Web usability and evaluation: Issues and concerns. In: Aykin N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, p. 243–249.

Beelders T. R., Blignaut P. J., McDonald T., Dednam E. (2007) The Impact of Different Icon Sets on the Usability of a Word Processor. Lect. Notes in Computer Science, vol. 4559, Springer, Berlin, Heidelberg, p. 250–257.

Böcker M., Larsson K. L., von Niman B. (2007) Increasing the Usability of Text Entry in Mobile Devices for European Languages and Languages Used in Europe. Aykin N. (Ed.) Usability and Internationalization. Global and Local User Interfaces, pt. 2, proc. Lecture Notes in Computer Science, vol. 4560, Springer Berlin / Heidelberg, p. 13–21.

Böcker M., Larsson K. L., von Niman B. (2006) Standardization Supporting Cultural Diversity: Character Repertoires, Ordering and Assignment to the 12-key Telephone Keypad for European Languages and Languages Used in Europe. Proceedings of the 8th conference on Human-computer interaction with mobile devices and services, ACM International Conference Proceeding Series, vol. 159, p. 281–29.

Boswell D., King B., Oescbger I., Collins P., Murphy E. (2002) Creating Applications with Mozilla. O'Reilly & Associates.

Boitet Ch. (2005) Message Automata for messages with variants, and methods for their translation. LNCS 3406.

Bowker L. (2005) Productivity vs Quality? A pilot study on the impact of translation memory systems. Localisation focus. The international journal for Localisation. Vol. 4, Issue 1, p. 13–20.

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.

Van de Burgt S. P., Tilanus P. A. J. (1989) Attributed ASN.1. In: Proceedings of the 2nd International Conference on Formal Description Techniques, FORTE 89, Amsterdam, p. 298–310.

Carey J. (1998) Creating global software: A conspectus and review. Interacting with Computers, Special Issue: Shared values and shared interfaces, 9, 4, p. 449–465.

Crimi C., Guercio A., Pacini G., Tortora G., Tucci M. (1990) Automating visual language generation. IEEE Transactions on Software Engineering, Vol. 16, Iss. 10, p. 1122–1135.

Crystal D. (2001) Language and The Internet. Cambridge: University Press.

Callahan E. (2006) Cultural Similiarities and Differencies in the Desin University Web sites. Journal of Computer-Mediated Communication, Vol. 11, No. 1, pp. 239–273.

Chavan A. L. (2007) Smart strategies for creating culture friendly products and interfaces. In: Aykin N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, p. 27–32.

Chen C.-H., Tsai C.-Y. (2007) Designing user interfaces for mobile entertaining devices with cross-cultural considerations. In: Aykin N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, p. 37–46.

Choong Y.-Y., 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.

Collins R.W. (2002) Software localization for Internet software, issues and methods. IEEE software, 19 (2): 74.

Davis M. (2007) Locale data markup language (LDML). Unicode Technical Standard #35. Unicode, Inc., http://unicode.org/reports/tr35/tr35-8.html [žiūrėta 2010-06-29]

Deitsch A, Czarnecki D. (2001) Java Internationalization. O'Reilly and Associates, 2001.

Dagienė V. (1998) Šiuolaikinės informacinės technologijos švietime: kalbos problema. Lituanistika pasaulyje šiandien: darbai ir problemos 3 d., p. 55–64. Vilnius: Baltos lankos.

Dagienė V. (2000) Lietuvių kalbos problema informacijos technologijos ir mokyklos sandūroje. XI pasaulio lietuvių mokslo ir kūrybos simpoziumas. Vilnius, 2000 m. birželio 21–26 d., p. 165–166.

Dagienė V., Grigas G. (2006) Quantitative evaluation of the process of open source software localization. Informatica vol.17, no.1, p. 3-12.

Dagienė V., Grigas G., Jevsikova T. (2004) Programinės įrangos lietuvinimas: patirties analizė. Informacijos mokslai. 31, p. 171–185.

Dagienė V., Grigas G., Jevsikova T. (2008) Enciklopedinis kompiuterijos žodynas. Vilnius: TEV.

Dagienė V., Jevsikova T. (2005) Virtualiosios mokymosi aplinkos lokalizavimo požiūriu Lietuvos matematikos rinkinys. ISSN 0132-2818. T. 45, spec. Nr., p. 197–202.

Dagienė V., Jevsikova T. (2009) Cultural Elements in Internet Software Localization. In: Lenca P., Brézillon P., Coppin G. (eds.) Revue d'intelligence artificielle. Human-centered processes – Current trends. Volume 23 – no 4/2009. Hermes – Lavoisier.

Dagienė V., Laucius R. (2004) Internationalization of open source software: Framework and some issues. In: Boyle T.; Oriogun P., Pakstas A. (Eds.) Proc. 2nd International Conference Information Technology - Research and Education (ITRE 2004), p. 204–207.

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

Deitsch A., Czarnecki D. (2001) Java Internationalization. O'Reilly, 446 p.

Deransart P., Maluszynski J. A. (1993) Grammatical View of Logic Programming. The MIT Press, Cambridge, Mass.

Drepper U., Meyering J., Pinard F., Haible B. (2010) GNU gettext tools, version 0.18. Native Language Support Library and Tools. Free Software Foundation.

DTK Computer (1991) Microsoft MS-DOS User's Guide and Refertence. Microsoft Corporation.

Dürst M. J. (2003) Internationalization of XML – Past, Present, Future. XML Conference&Exposition. Pensylvania, December 7–12.

Evers V., Day D. (1997) The role of culture in interface acceptance, In: Human Computer Interaction: INTERAC T'97, Sydney, Chapman and Hall.

Ember C. R., Ember M. (1977) Anthropology. Prentice-Hall, Englewood Cliffs, NJ.

Esselink B. (2000) A practical guide to localization. John Benjamins.

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

Evers V. (2001a) Cultural aspects of user interface understanding. Doctoral dissertation. Institute of Educational Technology, The Open University.

Evers V. (2001b) Cross-Cultural Understanding of Graphical Elements on the DirectED Website. Smith A. (Ed.) Proc. of Ann. Workshop on Cultural Issues on HCI, Univ. of Luton.

Ford G., Gelderblom J. H. (2003) The effects of culture on performance achieved through the use of human-computer interaction. Proceedings of SAICSIT, p. 218–230.

Frimannsson A., Hogan J. M. (2005) Adopting Standards-based XML File Formats in Open Source Localisation. Localisation Focus. The International Journal for Localisation. Vol. 4, Issue 4, p. 9–23.

Faltstrom P., Hoffman P., Costello A. (2003) Internationalizing Domain Names in Application. Network Working Group, Request for Comments: 3490.

Frost R. A. (1992) Constructing programs as executable attribute grammars. The Computer Journal, Vol. 35, Iss. 4. Special issue on models and architectures, p. 376–389.

Gaspari F. (2007) The role of online MT in Webpage translation. A thesis for a degree of Doctor of Philosophy. Universtity of Manchester.

Golovinas B. (1982) Kalbotyros įvadas. Vilnius.

Gordon R. G. (2005) Ethnologue: Languages of the World. 15th edition, SIL International, 1272 p.

Gribbons W. (1997) Designing for the Global Community. Proceedings. of IEEE Intl. Professional Communication Conference, (Salt Lake City, US), p. 261–273.

Grigas G. (1998) Lietuviški rašmenys ir kodų lentelės. Informatika. Nr. 33, p. 21–46.

Grigas G., Pedzevičienė S. (2007) Asmenvardžių ir vietovardžių rašybos klaidos pokalbių programų registracijos duomenyse. Informacijos mokslai. 2007, t. 42–43, p. 141–144.

Grigas G., Pedzevičienė S. (2009) Klaviatūros įtaka rašybos klaidoms, susijusioms su rašto ženklų vartojimu. Informacijos mokslai, t. 50, p. 200–204.

Grigas G. (2000) Programavimo kalbų leksikos elementų analizė mokymo požiūriu. Informatika. 1(35), p. 45–67.

Grigas G., Zalatorius J. (2000) Cultural and economic aspects of software localization. Baltic IT Review, nr. 6, p. 61–64.

Grigas G. (2006) Santrumpos terminijoje. Terminologija, t. 13, p. 194–199.

Grigas G. (2008) Kompiuterijos leksika ir terminija. Santalka, t. 16, Nr. 2, p. 31–39.

Guseva N. (2009) Interneto svetainių adaptavimas kultūrai kaip kokybės gerinimo būdas. Ekonomika ir vadyba, 14, p. 1060–1068.

Guzman R. (2004) Tools Review: Catalyst and Terminology Wizard. Localisation focus. The international journal for Localisation. Vol. 3, Issue 1, p. 18–19.

Hall B. (2007) Resources in Microsoft .NET. Ccaps Newsletter, p. 1–6.

Hall E., Hall M. (1990) Understanding Culture Differences. Maine, Intercultural Press.

Hall P. V., Hudson R. (1997*)* Software without frontiers: A Multi-Platform, Multi-Cultural, Multi-Nation Approach. Willey & Sons, 350 p.

Hansmeyer C. (2002) The History of Phonetic Alphabets. University of Bremen, 15 p.

Hofstede G., Hofstede G. J. (2005) Cultures and organizations: software of the mind (Revised and expanded 2nd ed.). New York: McGraw-Hill.

Hogan J. M., Ho-Stuart C., Pham B. (2004) Key challenges in software internationalisation. In: Proceedings of the Australasian Workshop on Software Internationalisation (AWSI 2004). ACSW Frontiers 2004: Conferences on Research and Practice in Information Technology Volume 32, Sydney: Australian Computer Society, p. 187–194.

Hofmann P. (2007) Localising and internationalising graphics and visual information. IEEE Transactions on Professional Communication vol. 50, no. 2, p. 91–92.

Hofstede G. (1991) Cultures and Organization. McGraw Hill, New York.

Horwitz S., Reps T. (1992) The use of program dependence graphs in software engineering. In Proceedings of the 14th international conference on Software engineering. ACM New York, NY, USA, p. 392–411.

IBM (1994) National Language Design Guide Volume 1, National Language Support Reference Manual, 4th Ed.

IPA[1] (1999). The Handbook of the International Phonetic Association. Cambridge University Press, 214 p.

Yang Y. X. (2007) Extending the user experience to localized products. In: Aykin N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, p. 285–292.

Yijun Yu, Jianguo Lu, Mylopoulos J., Weiwei Sun, Jing-Hao Xue, D'Hollander E.H. (2005) Making XML document markup international. Software - Practice and Experience vol. 35, no. 1, p. 1–14.

Jevsikova T. (2003) Programų adaptavimas lietuviškai lokalei. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, ISBN 9955-09-335-8, Kaunas: Technologija, p. I-(8–14).

Jevsikova T. (2006) Internationalization and Localization of Web-based Learning Environment. In: R. Mittermeir (Ed.) Informatics Education – the Bridge Between Using and Understanding Computers. LNCS, 4226, p. 310–319.

Jevsikova T. (2009) Programinės įrangos lokalizavimas. Daktaro disertacija. Matematikos ir informatikos institutas, Vytauto Didžiojo universitetas.

Jones L. G., Simon J. (1986) Hierarchical VLSI design systems based on attribute grammars. In: Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, p. 58–69.

Kano N. (1995) Developing International Software: For Windows 95 and Windows NT, Microsoft, Redmond, WA, 743 p.

Kaiser G., Kaplan S. M. (1993) Parallel and distributed incremental attribute evaluation algorithms for multiuser software development environments. ACM Transactions on Software Engineering and Methodology. Vol. 2, Iss. 1, p. 47–92.

Kaulakienė A. (2000) Kompiuterijos terminijos sinonimija: yda ar būtinybė. Terminologija t. 6, p. 23–28.

Kaulakienė A. (2006) Naujausios technologijos ir kompiuterijos terminija. Santalka. t. 14, nr. 4, p. 24–27.

Kersten G. E., Kersten M. A., Rakowski W. M. (2002) Application Software and Culture: Beyond the Surface of Software Interface. Journal of Global Information Management, InterNeg Reports INR1/01.

Kersten G. E., Matwin S., Noronha S. J., Kersten M. A. (2000) The Software for Cultures and the Cultures in Software. Proceedings of the 8th European Conference on Information System. ECIS 2000, Hansen H.R., Bichler M., Harald H. (Eds.). Vienna University of Economics and Business Administration. Vol. 1, p. 509–514.

Knuth D. E. (1968) Semantics of context-free languages. Theory of Computing Systems, vol. 2, no. 2, p. 127–145.

Kondratova I., Goldfarb I. (2006) Cultural Interface Design: Global Colors Study. Lect. Notes in Computer Science, vol. 4277, Springer, Berlin, Heidelberg, pp. 926–934.

Korpela J. (2006) Character histories: notes on some Ascii code positions, http://www.cs.tut.fi/~jkorpela/latin1/ascii-hist.html

Lanier C. R. (2005) Linux and the appeal to cultural values. IEEE Technology and Society Magazine vol. 24, no. 4, p. 12–17, 42.

Laucius R. (2003) Lokalės, jų sandara ir ypatumai. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, ISBN 9955-09-335-8, Kaunas: Technologija, p. I-(1–7).

Laucius R. (2007) Kompiliatorių internacionalizacija. Daktaro disertacija. Technologijos mokslai, informatikos inžinerija (07T). Vilniaus Gedimino technikos universitetas, Matematikos ir informatikos institutas, Vilnius.

Laucius R., Dagienė V. (2003) Raštinės programinės įrangos „OpenOffice.org“ adaptavimas lokalės normoms. Informacijos mokslai, 26, p. 240–245.

Laucius R. (2005) *Free Pascal* kompiliatoriaus internacionalizavimas. Informacijos mokslai, 34, p. 302–306.

Lewi J., De Vlaminck K., Steegmans E., Van Horebeek J. (1992) Software Development by LL(1) Syntax Description. John Wiley & Sons, New York.

Lepouras G., Weir G.R.S. (2003) Subtitled interaction: complementary support as an alternative to localization. International Journal of Human-Computer Studies vol.59, no.6 : p. 941–957.

Marcus A., Gould E. W. (2000) Crosscurrents: cultural dimensions and global Web user-interface design. ACM Interactions 7(4), p. 32–46.

Marcus A., Gould E. W. (2001) Cultural dimensions and global Web design. Experience Intelligent design. AM+A.

Maroto J., Bortuli M. (2001) Web site localization. Proceedings of the European Languages and the Implementation of Communication and Information Technologies (Elicit) conference. University of Paisley.

Melby A. K., Warner T. C. (1995) The Possibility of Language: A Discussion of the Nature of Language with Implications for Human and Machine Translation. Amsterdam/Philadelphia: John Benjamins.

Microsoft Corporation (1990). Microsoft windows international handbook for software design, Microsoft Corporation, Redmond, Washington.

Moore L. (2006) CLDR: The common locale data repository – locales for the world. In: Kelly K., Schaler R. (Eds.) The 11th annual internationalisation and localisation conference organised by the Localisation research centre. The Localisation factory. 25–26 October, 2006, Dublin, Ireland. p. 31–43.

Manousopoulou A. G., Papakonstantinou G. (2000) A Grammatical Approach to User Model Resolution for Electronics. In: Proceedings of the Third Workshop on Attribute Grammars and their Applications, p. 117–124.

Mushtaha A., De Troyer O. (2005) Towards Localising e-Learning Websites. Localisation Focus. The International Journal for Localisation. Vol. 4, Issue 4, p. 6–8.

Musale S. (2004) Getting more from translation memory. Localisation focus. The international journal for Localisation. Vol. 3, Issue 1, p. 9–10.

Neven F. (2000) Extensions of Attribute Grammars for Structured Document Queries. Research Issues in Structured and Semistructured Database Programming, LNCS, Vol. 1949/2000, p. 99–117.

Nielsen J. (1990) (Ed.) Designing User Interfaces for International Use, Elsevier Science, Amsterdam.

Nichols D. M., Witten I.H., Te Taka Keegan, Bainbridge D., Dewsnip M. (2005) Digital libraries and minority languages. New Review of Hypermedia and Multimedia vol.11, no. 2: p. 139–155.

Oškinis A. (1991) Kompiuteris ir lietuvių kalba. Kompiuteriai. Nr. 1, 5–9.

Paaki J. (1995) Attribute Grammar Paradigms — A High-Level Methodology in Language Implementation. ACM Computmg Surveys, Vol. 27, No, 2.

Palionis J. (1985) Kalbos mokslo pradmenys. Vilnius, „Mokslas“.

Psaila G., Crespi-Reghizzi S. (1999) Adding Semantics to XML. In: D. Parigot and M. Mernik (eds.), Procedings of the Second Workshop on Atribute Grammars and their Applications, Amsterdam, The Netherlands, p. 113–132.

Qingxin Shi (2007) Cultural Usability: The Effects of Culture on Usability Testing. Human-Computer Interaction – INTERACT 2007, LNCS, 4663, p. 611–616.

Ralys D., Dadurkevičius V. (1991) Lietuviški rašmenys kompiuteryje. Mokslas ir technika, Nr. 5, p. 29-31.

Reinecke K., Bernstein A. (2007) Culturally adaptive software: Moving beyond internationalization. In: Aykin N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, p. 201–210.

Roach P. (2000) English Phonetics and Phonology. Cambridge University Press, Cambridge, 192 p.

Russo P., Boor S. (1993) How fluent is your interface? Designing for international users, In INTERCHI'93 Human Factors in Computing Systems, Amsterdam, NY: ACM.

Reghizzi S. C. (2009) Formal Languages and Compilation. Texts in Computer Science. Springer.

Ramalho J. C., Lopes, A. R. (1998) Henriques P. R. Generating SGML specific editors: from DTDs to Atribute Grammars. In: Markup Technologies Conference, Chicago.

Sandrini P. (2005) Website Localization and Translation. MuTra 2005 – Challenges of Multidimensional Translation: Conference Proceedings, p. 1–8.

Sapir E. (1949) Selected Writings of Edward Sapir. University of California Press, Berkeley.

Savourel Y. (2001) XML Localization. SAMS, 519 p.

Scattergard D. (2002) Documentation Localisation Costs. Localisation Focus, vol. 1, issue 2.

Schäler R. (2002) The Cultural Dimensions in Software localization. Localisation Focus, vol. 1, isssue 2.

Shen J. (2000). User Interface Internationalization. CIS732 Final Project.

Shinoda Y., Katayama T. (1998) Attribute grammar based programming and Its environment. In: proceedings of the 21st Hawaii International Conference on System Sciences. IEEE Computer Society Press, p. 612–620.

Schadewitz N., Jachna T. (2007) Introducing new methodologies for identifying design patterns for internationalization and localization. In: Aykin N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, p. 228–237.

Schwartz S. (1994) Beyond individualism/collectivism: New cultural dimensions of values. In: U. Kim, H. C. Triandis, C. Kigitcibasi, S., C. Choi., & G. Yoon (Eds.), Individualism and Collectivism: Theory, Method, and applications, pp.85-119). Thousand Oaks, CA: Sage.

Singh N. (2002) Analyzing Cultral Sensitivity of Websites: A Normative Framework. Journal of Practical Global Business, vol. 1, No. 2, pp. 32–53.

Singh N., Zhao H., Hu X. (2003) Cultural Adaptation on the Web: A Study of American Companies' Domestic and Chinese Websites. Journal of Global Information Management, Vol. 11, No. 3, pp. 63–80.

Sližienė N., Valeckienė A. (sud.) (1992) Lietuvių kalbos rašyba ir skyryba.

Sproat R. (2003) A Formal Computational Analysis of Indic Scripts. International Symposium on Indic Scripts: Past and Future, Tokyo.

Somers H. (2003) Translation Memory Systems. In: Somers, H. (ed.) Computers and Translation: A Translator's Guide. Amsterdam/Phyladelphia: John Benjamins, p. 31–46.

Souchon N., Vanderdonckt J. (2003) A Review of XML-compliant User Interface Description Languages, LNCS 2844.

O'Sullivan P. (2001) A Paradigm for Creating Multilingual Interfaces. Doctoral Dissertation. University of Limerick.

O'Sullivan P. (1999) User Interface and Translation. Results of a survey completed for Lotus Development Communication Products group.

Taylor D. (1992) Global Software: Developing Applications for the International Market. Springer-Verlag New York.

Teasley B., Leventhal L., Blumenthal B., Instone K. and Stone D. (1994) Cultural diversity in user interface design: Are intuitions enough? SIGCHI Bulletin, 26, 1, p. 36–40.

Trompenaars F. (1997) Riding the waves of culture – Understanding cultural diversity in business. Nicholas Brealey Publishing, London.

Tumasonis V., Grigas G. (2000). Naujosios lietuviškos kompiuterio klaviatūros sandara. Informacijos mokslai. t. 14, p. 105–112.

Tumasonis V. (2007) Lietuviškos kirčiuotos raidės: kodavimas ir įvedimas iš klaviatūros. Informacijos mokslai. t. 42–43, p. 135–140.

Tuoc V., David J. T., Driscoll K. (1995) Internationalization: Developing Software for Global Markets. John Wiley & Sons, Inc.

The Unicode Standard (2006) Version 5.0. Addison-Wesley Professional.

Uren E., Howard R., Perinotti T. (1993) Software internationalization and localization. An Introduction. Van Nostrand Reinhold.

Vaske J., Grantham C.E. (1990) Socializing the Human-computer Environment. Ablex, Norwood, NJ.

Vatrapu R. (2002). Culture and International Usability Testing: The effects of Culture in Structured Interviews. Master thesis,Virginia Polytechnic Institute and State University.

Vitkauskaitė E. (2009) Kultūrinių skirtumų įtaka tinklalapių turiniui. Ekonomika ir vadyba, t. 14, pp. 541–552.

Voegelin C. F., Voegelin F. M. (1977) Classification and index of the world's languages. New York: Elsevier.

Wassmer T. (2004) Comparing Tools Used in Software Localization. Localisation reader, 2003–2004, p. 17–21.

Wan Mohd Isa W. A. R., Md Noor N. L. (2007) Incorporating the cultural dimensions into the theoretical framework of website information architecture. In: Aykin N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, p. 212–221.

Yeo A. (1996) World-wide CHI: Cultural user interfaces, a silver lining in cultural diversity. SIGCHI, 28(3), p. 4–7.

Zeller I. (2006) Lokalizacijos iššūkiai. Darbai ir dienos, t. 45, p. 79–96.

Zerfass A. (2005) TMX and SRX Exchanging TM Data. In: The Global Initiative for Local Computing. Proc. of 10th Annual Internationalization and Localization Conference Organised by the LRC (LRC-X). LRC, Limerick, p. 131–141.

Zinkevičius Z. (1980) Kalbotyros pradmenys. Kaunas.

Žalkauskas V. (2003) Šiuolaikinių kompiuterių programų ir tinklų žodynas. MELI.

Žalkauskas V. (2004a) „Microsoft Windows" žodynėlis. Smaltija.

Žalkauskas V. (2004b) „Linux" žodynėlis. Smaltija.

Žalkauskas V. (2005) Biuro programų žodynėlis. MELI.

Standartai

ASCII:1986. American National Standard for Information Systems — Coded Character Sets — 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986, American National Standards Institute, Inc.

ECMA-6 (1997). 7-bit Coded Character Set.

ETSI ETS 300 640 (1996). Human Factors (HF); Assignment of alphabetic letters to digits on Standard telephone keypad arrays. Human Factors (HF). User Interfaces. Character repertoires, orderings and assignments to the 12-key telephone keypad (for European languages and other languages used in Europe).

ETSI ES 202 130 V1.1.1 (2003). Human Factors (HF). User Interfaces. Character repertoires, orderings and assignments to the 12-key telephone keypad (for European languages and other languages used in Europe).

ETSI ES 202 130 V2.1.2 (2007). Human Factors (HF). User Interfaces. Character repertoires, orderings and assignments to the 12-key telephone keypad (for European languages and other languages used in Europe).

ISO 31-1:1992. Quantities and units – Part 1: Space and time.

ISO 639-1:2002. Codes for the representation of names of languages – Part 1: Alpha-2 code.

ISO 639-2:1998. Codes for the representation of names of languages – Part 2: Alpha-3 code.

ISO/IEC 646:1991. Information technology — ISO 7-bit coded character set for information interchange.

ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions – Part 1: Country codes.

ISO 8601:2004. Data elements and interchange formats – Information interchange – Representation of dates and times.

ISO/IEC 8859. Information technology — 8-bit single-byte coded graphic character sets. Part 1 – Part 16.

ISO 8879:1986: Information processing. Text and office systems. Standard Generalized Markup Language (SGML).

ISO/IEC 9899:1999. Programming languages. C.

ISO/IEC 9945-1:2003 Information technology – Portable Operating System Interface (POSIX) – Part 1: Base Definitions.

ISO/IEC 9945-2:2003. Information technology – Portable Operating System Interface (POSIX). Part 2: System Interfaces.

ISO/IEC/IEEE 9945:2009. Information technology – Portable Operating System Interface (POSIX®) Base Specifications, Issue 7.

ISO/IEC 9995. Information technology – Keyboard layouts for text and office systems. Part 1 – Part 8.

ISO/IEC 10646-1:1993. Information Technology – Universal Multiple-Octet Coded Character Set (UCS-2) – Part 1: Architecture and Basic Multilingual Plane.

ISO 12199:2000. Alphabetic ordering of multilingual terminological and lexicographical data represented in the Latin alphabet.

ISO/IEC 14651:2007. Information technology – International string ordering and comparison – Method for comparing character strings and description of the common template tailorable ordering.

ISO/IEC 14652:2004. Information technology – Specification method for cultural conventions.

ISO/IEC 14882:2003. Programming languages. C++.

ISO/IEC 15897:1999. Information technology – Procedures for registration of cultural elements.

ISO 15924. Codes for the representation of names of scripts.

ISO 16642:2003. Computer applications in terminology – Terminological markup framework.

ISO 30042:2008. Systems to manage terminology, knowledge and content – TermBase eXchange (TBX).

ITU-T Recommendation E.161 (2001). Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network.

LST 1093-89. Informacijos apdorojimo sistema. Daugiakalbis koduotų ženklų rinkinys su lietuviškomis raidėmis. Aštuonių bitų kodai.

LST 1095-89. Informacijos apdorojimo sistema. Personalinių kompiuterių ženklų rinkinys su lietuviškomis raidėmis. Aštuonių bitų kodai.

LST 1282:1993. Informacijos technologija. 8 bitų lotyniškos abėcėlės koduotų ženklų rinkinys, papildytas lietuviškos abėcėlės raidėmis, skirtas naudoti WINDOWS, UNIX aplinkoje.

LST 1283:1993. Informacijos technologija. 8 bitų lotyniškos abėcėlės koduotų ženklų rinkinys, papildytas lietuviškos abėcėlės raidėmis, skirtas naudoti MS DOS aplinkoje. 774 koduotė.

LST 1284:1993. Informacijos technologija. 8 bitų lotyniškos abėcėlės koduotų ženklų rinkinys, papildytas lietuviškos ir rusiškos abėcėlės raidėmis, skirtas naudoti MS DOS aplinkoje. 772 koduotė.

LST 1564:2000. Informacijos technologija. Ženklų kodavimas 8 bitais. Lietuviškų kirčiuotų raidžių rinkinys.

LST 1582:2000. Informacijos technologija. Lietuviška kompiuterio klaviatūra. Ženklų išdėstymas.

LST 1590-1:2000. Informacijos technologija. Ženklų kodavimas 8 bitais. 1 dalis. Grafinių ženklų rinkinys DOS terpei.

LST 1590-2:2000. Informacijos technologija. Ženklų kodavimas 8 bitais. 2 dalis. Lietuviškų kirčiuotų raidžių ir transkripcijos ženklų rinkinys DOS terpei.

LST 1590-3:2000. Informacijos technologija. Ženklų kodavimas 8 bitais. 3 dalis. Grafinių ženklų rinkinys Windows terpei.

LST 1590-4:2000. Informacijos technologija. Ženklų kodavimas 8 bitais. 4 dalis. Lietuviškų kirčiuotų raidžių ir transkripcijos ženklų rinkinys Windows terpei.

LST ISO 8601:1997. Duomenų elementai ir pasikeitimo informacija formatai. Pasikeitimas informacija. Datų ir laiko žymėjimas.

LST ISO/IEC 8859-13:2000. Informacijos technologija. 8 bitais koduotų ženklų rinkiniai. 13 dalis. Lotynų 7-oji abėcėlė (tapatus ISO/IEC 8859-13:1998).

LST ISO/IEC 10646-1:2000. Informacijos technologija. Universalus keliais bitais koduotų ženklų rinkinys. 1 dalis. Sandara ir pagrindinė daugiakalbė lentelė.

LST ISO/IEC 15897:2001. Informacijos technologija. Kultūros elementų registravimo procedūros (tapatus ISO/IEC 15897:1999).

OASIS (2008). XLIFF version 1.2. OASIS Standard.

RFC 1034 (1987). Domain names – concepts and facilities. P. Mockapetris.

RFC 2045 (1996) . Multipurpose Internet Mail Extensions (MIME). Part One. Format of Internet Message Bodies. N. Freed, N. Borenstein.

RFC 2046 (1996). MIME Part Two: Media Types. N. Freed, N. Borenstein.

RFC 2047 (1996). MIME Part Three: Message Header Extensions for Non-ASCII Text. K. Moore.

RFC 2048 (1996). Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures. N. Freed, J. Klensin, J. Postel.

RFC 2049 (1996). Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. N. Freed, N. Borenstein.

RFC 3066 (2001). Tags for the Identification of Languages, H. Alvestrand.

RFC 3492 (2003). Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA), A. Costello.

Dalykinė rodyklė

F

G

H

I

Y

J

K

L

M

N

O

P

R

S

Š

T

U

V

W

X

Ž

Dagienė, Valentina

Da77 Programinės įrangos lokalizavimas / Valentina Dagienė, Gintautas Grigas, Tatjana Jevsikova. – Vilnius : Matematikos ir informatikos institutas, 2010. – 328 p. : iliustr.

Santr. angl. – Bibliogr.: p. 307–314 ir išnašose. – Dalyk. r-klė: p. 319–327.

ISBN 978-9986-680-47-5

Monografijoje nagrinėjama programinės įrangos lokalizavimo eiga, pagrindiniai komponentai, įvardijamos problemos, teikiami siūlymai joms spręsti, aptariamos kultūrinės dimensijos.

Monografija visų pirma skiriama lokalizavimo problemas tyrinėjantiems mokslininkams, lokalizavimo arba internacionalizavimo tematiką pasirinkusiems doktorantams. Knyga turėtų dominti ir lokalizuotojus praktikus – pagelbėtų geriau suvokti lokalizavimo procesus, našiau ir sistemingiau atlikti lokalizavimo darbus.

UDK 004.41

Software localisation

The monograph covers the process of software localisation, its main components, discusses the problems, arising during localisation, proposes their possible solutions, and deals with cultural dimensions.

The monograph is intended for scientists and researchers in the field of software localisation, as well as for PhD students who have chosen the topics of software localisation and internationalisation. The book is also supposed to interest practical software localisers. It should help them understand and organize localisation processes better, accomplish localization tasks in a more effective way.

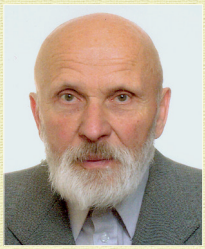
Valentina Dagienė, Gintautas Grigas, Tatjana Jevsikova
PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Monografija

Dalykinis redaktorius *Viktoras Dagys*
Iliustracijas parengė *Tatjana Jevsikova*, *Rimantas Žakauskas*
Viršelio autorius *Aidas Žandaris*
Maketavo *Rasa Ališauskienė*

Formatas 70×100/16, 15,85 aut. l., 20,5 apsk. leid. l.
Tiražas 200 egz.
Išleido Matematikos ir informatikos institutas, Akademijos g. 4, LT-08663 Vilnius.
Interneto svetainė: www.mii.lt
Parengė spaudai leidykla „Žara“, a. d. 2699, LT-03007, Vilnius
Interneto svetainė: www.žara.lt
Spausdino KTU spaustuvė, Studentų g. 54, LT-51424 Kaunas

Valentina Dagienė
Informatikos daktarė, atliko edukologijos habilitaciją, profesorė. Parašė 60 knygų, paskelbė per 250 mokslinių ir metodinių straipsnių, vadovavo per 30 projektų. Lietuvos mokslo premijos (informatikos mokymas, mokomosios programinės įrangos kūrimas ir lokalizavimas) laureatė.



Gintautas Grigas
Technikos mokslų daktaras, docentas, projektavęs pirmąjį lietuvišką kompiuterį „Rūta“. Programavimo mokymo Lietuvoje ir programinės įrangos lietuvinimo pradininkas. Parašė 35 knygas iš kompiuterijos ir programavimo, paskelbė per 300 mokslinių ir metodinių straipsnių.

Tatjana Jevsikova
Informatikos daktarė, disertaciją apgynusi iš programinės įrangos lokalizavimo (vadovė V. Dagienė). Lokalizavo per 10 įvairių kompiuterių programų, domisi elektroninio mokymosi sistemomis, kompiuterinėmis mokomosiomis programomis.

9 789986 680468