Lietuvos matematikos rinkinys. LMD darbai
51 tomas, 2010, 85–90

ISSN 0132-2818
www.mii.lt/LMR/

Programavimo stilius ir programų internacionalizavimo mokymas

Viktoras Dagys, Gintautas Grigas

Matematikos ir informatikos institutas
Akademijos g. 4, LT-08663 Vilnius
E. paštas: dagys@ktl.mii.lt; grigas@ktl.mii.lt

Santrauka. Nagrinėjamos galimybės supažindinti mokinius su programinės įrangos internacionalizavimo pradmenimis. Parodoma, kad pagrindines žinias galima pateikti nesudėtingų praktinių programavimo užduočių pavidalu, ypač jeigu prieš tai mokant programavimo buvo kreipiamas dėmesys į mokinio rašomų programų tekstą, supažindinama su programavimo stiliaus elementais. Pateiktoji tema yra daugiatikslė: mokinys įgytų supratimą apie programų internacionalizavimą ir lokalizavimą, pakankamai žinių, kad galėtų testuoti praktiškai naudojamas, ypač atvirąsias, programas (tai galėtų būti rimta motyvacija), susipažintų su pasaulio kalbų įvairove, giliau suvoktų lietuvių kalbos reikšmę ir jos vietą tarp kitų kalbų.

Raktiniai žodžiai: programavimo mokymas, programų internacionalizavimas, programų lokalizavimas, programavimo stilius.

1 Įvadas

Programinės įrangos lokalizavimo darbų apimtis nuolat didėja. Tai lemia programinės įrangos gamybos globalizacija: programa, pagaminta vienoje valstybėje eksportuojama į daugelį kitų valstybių. Tuo pačiu ir verčiama (lokalizuojama) į daugelį kalbų [1]. Lietuvoje taip pat didėja lokalizavimo darbų, nes vis daugiau programų reikia pritaikyti darbui lietuviškoje aplinkoje. Kita vertus, būtų naudinga, kad Lietuvoje sukurtas programas būtų galima eksportuoti. O tam reikia, kad jos būtų internacionalizuotos. Deja, programinės įrangos internacionalizavimo ir lokalizavimo studijoms dar nekreipiamas reikiamas dėmesys. Mūsų žiniomis, lokalizavimas dėstomas tik trijose mokymo įstaigose.

Vilniaus universitete lokalizavimas dėstomas bakalauro studijų matematikos ir informatikos mokymo programos studentams pagal Valentinos Dagienės ir Rimgaudo Lauciaus parengtą programą [5, 4]. Būsimi informatikos mokytojai naudosis lokalizuota programine įranga, o kai kurie gal ir prisidės prie jos lokalizavimo, todėl supratimas apie lokalizavimą jiems reikalingas. Patys jie vargu ar kurs programas, juo labiau eksportuotinas, tad su internacionalizavimu supažindinami epizodiškai.

Kauno technologijos universitete lokalizavimas dėstomas humanitarams – technikos kalbos vertėjams magistrantams.

Matematikos ir informatikos institute lokalizavimas dėstomas doktorantams, tačiau nereguliariai, kai būna doktorantų, kurių temos susijusios su lokalizavimu ar internacionalizavimu. Maždaug trečdalis kurso skirta internacionalizavimui.

Taigi internacionalizavimo mokymas dar epizodiškas, o programuotojai, kuriems šios žinios daugiausiai reikalingos, su juo net nesupažindinami. Baigusiems studijas

žinių spragas tenka užpildyti savarankiškai arba daryti neinternacionalizuotas programas.

Pirmąją pažintį su programavimu galima įgyti vidurinėje mokykloje pasirinkus informacinių technologijų išplėstinio kurso programavimo modulį. Yra entuziastų, kurie savarankiškai pradeda programuoti arba lokalizuoti programas dar besimokydami vidurinėje mokykloje, tačiau tai tik vienetai. Yra gerų vadovėlių (pvz., [2]), tačiau jie skirti profesionalams ir pradiniam susipažinimui pernelyg sudėtingi.

Bendrojo lavinimo mokyklų mokiniai gali įgyti papildomų programavimo žinių mokydamiesi Jaunųjų programuotojų neakivaizdinėje mokykloje. Šioje mokykloje kreipiamas dėmesys programavimo kultūrai, programavimo stiliui, vertinamas pats programos tekstas, o ne vien jos darbo rezultatai. Tai sudaro pagrindą internacionalizavimo elementams mokyti.

2 Ryšys tarp programos modifikavimo, internacionalizavimo ir koregavimo

Viena iš svarbių programos savybių yra galimybė ją modifikuoti – pakoreguoti taip, kad ji tiktų prie kitų, pasikeitusių sąlygų, išaugusių naudotojo poreikių ir pan. Programos lokalizavimas yra jos pritaikymas kitai kalbos ir kultūros terpei. Taigi jį galima laikyti atskiru programos modifikavimo atveju.

Kuriant programą paprastai pasirūpinama ir būsimo jos modifikavimo, tobulinimo galimybėmis: programa struktūrinama, pasikartojantys arba autonominiai veiksmai iškeliami į funkcijas ir procedūras, parametrizuojamas duomenų perdavimas tarp jų, tai, kas numatoma modifikuoti (skaitiniai parametrai, naudojamų duomenų failų vardai ir pan.), iškeliama į programos pradžią, dažniausiai į konstantų skyrių, stengiamasi, kad programos tekstas būtų lengvai skaitomas ir suprantamas. Jeigu į visa tai buvo kreipiamas dėmesys mokant programavimo pradmenų, besimokančiajam bus nesunku suvokti ir pagrindinius internacionalizavimo elementus. Tai suvokęs ir laikydamasis tam tikrų (internacionalizavimo) taisyklių galės programą iš pat pradžių projektuoti taip, kad nesunkiai būtų gaunamas rezultatas – internacionalizuota programa.

3 Tekstų eilutės

Pagrindinis internacionalizavimo uždavinys – lokalizuojamuosius išteklius (dažniausiai – tekstus, kuriuos reikia išversti) atskirti nuo programos vykdomosios dalies. Tai parodysime, pateikę elementarių programų pavyzdžių.

1 pavyzdys.

```
program dalikliai_1; {Kiek daliklių turi skaičius}
 var n,   {pradinis duomuo}
      dal, {rezultatas − daliklių skaičius}
      i: integer;
begin
 writeln('Įveskite skaičių');
 readln(n);
 writeln('DALIKLIŲ SKAIČIUS');
 writeln('Pradinis duomuo: ', n);
```

```
  dal := 1;
  for i := 1 to n div 2 do
    if n mod i = 0 then dal := dal + 1;
  writeln('Rezultatas: ', dal);
end.
```

Tai neinternacionalizuota programa. Tekstai, kurie vykdant sukompiliuotą programą bus matomi kompiuterio ekrane, išbarstyti po visą programą. Lokalizavimo metu juos reikia rasti ir išversti į kitą kalbą, o po to programą iš naujo kompiliuoti. Kai programa nedidelė, tai nesunku padaryti, bet kai didesnė, gerokai sunkiau rasti visus lokalizuotinus tekstus, galima suklysti išvertus ne tai, ką reikia (pvz., kintamojo vardą).

Pirmas žingsnis į internacionalizavimą būtų surinkti visus verstinus tekstus į vieną vietą – konstantų dalį, atskirai nuo kitų nelokalizuojamų tekstų (kintamųjų vardų, komentarų). Lokalizuoti bus patogiau, bet programą vis tiek reikės perkompiliuoti. Norint perkompiliavimo išvengti tenka lokalizuotinus išteklius iškelti į atskirą failą. Pavadinkime jį tekstai.txt ir pakoreguokime programą.

2 pavyzdys.

```
program dalikliai_2;
  var n,   {pradinis duomuo}
      dal, {rezultatas − daliklių skaičius}
      i, j: integer;
      tekstai: text; {lokalizuojamieji ištekliai}
      e: array [1..4] of string;
begin
  assign(tekstai, 'tekstai.txt'); reset(tekstai);
  for j := 1 to 4 do readln(tekstai, e[j]);
  writeln(e[1]);
  readln(n);
  writeln(e[2]);
  writeln(e[3], ' ', n);
  dal := 1;
  for i := 1 to n div 2 do
    if n mod i = 0 then dal := dal + 1;
  writeln(e[4], ' ', dal);
end.
```

Išteklių failas *tekstai.txt*

```
Įveskite skaičių
DALIKLIŲ SKAIČIUS
Pradinis duomuo
Rezultatas
```

Čia lokalizuojamieji ištekliai atskirti nuo vykdomosios programos dalies. Norint programą lokalizuoti pakanka išversti failą *tekstai.txt*.

Realiose programose būna šimtai ir tūkstančiai lokalizuojamų eilučių. Naujose programų laidose eilučių skaičius keičiasi (vienos išbraukiamos, kitos įdedamos), todėl išteklių failuose jos identifikuojamos numeriais arba vardais ir nebereikia rūpintis eilučių išdėstymo tvarka.

4 Parametrizuotos eilutės

Ekrane rodomos eilutės dažnai formuojamos iš atskirų frazių. Pavyzdžiui, minėtos programos pranešimai apie duomenis ir rezultatus būtų sklandesni juos sujungus į vieną rišlų sakinį: Skaičius *n* turi *dal* daliklį(-ius, -ių)

Tokį tekstą galima suformuoti tiesiog išvedimo sakinyje:

```
writeln('Skaičius ', n, 'turi ', dal, ' daliklį(-ius, -ių)')
```

arba

```
writeln(e11, n, e12, dal, e13)
```

Tačiau atskirus žodžius, nematant juos supančio konteksto, būtų kebliau versti. Galima suklysti parenkant jų gramatines formas. Be to, sakiniuose kitomis kalbomis žodžių tvarka gali būti kitokia. Šių trūkumų išvengiama naudojant parametrus. Tokie tekstai lokalizuotinuose ištekliuose paprastai pateikiami viena parametrizuota eilute: Skaičius %1 turi %2 daliklį(-ius, -ių)

Procento ženklas – tai informacija programai, kad po jo eina parametras (pavyzdyje parametrai įvardinti skaičiais). Lokalizuojant tokią eilutę parametrus galima įterpti ten, kur reikia.

Besimokančiajam, susipažinusiam su funkcijų ir procedūrų parametrais, tai nebus naujiena, tačiau reikės programuoti veiksmus su eilutėmis. Tai daugeliui gali būti neįprasta, nes programavimo kursuose darbui su eilutėmis skiriama mažai dėmesio. Internacionalizavimo tematika paskatintų giliau susipažinti su eilučių operacijomis, kas aktualu ne vien čia. Programavimo darbų, skirtų tekstams apdoroti, apimtys auga sparčiau, negu skirtų apdoroti skaičiams.

5 Dinaminės eilutės

Liko išspręsti dar vieną uždavinį: žodžio *daliklis* gramatinę formą priderinti prie kintančios antrojo parametro reikšmės ir ekrane rodyti tik tuo momentu tinkamą. Tai mokiniams pažįstamas uždavinys. Tuos, kurie jį programavo, galima laikyti jau netiesiogiai susipažinusiais su vienu programų internacionalizavimo elementu.

Internacionalizuotoje programoje žodžio formos nustatymą, priklausomai nuo jo parametro reikšmės, reikia išskirti į atskirą funkciją (procedūrą), dar geriau – į atskirai kompiliuojamą modulį, kad jį lokalizavimo metu būtų galima pakeisti tinkamu konkrečiai kalbai, nes skiriasi formų skaičius (jų būna nuo 1 iki 6) ir jų nustatymo algoritmai. Pavyzdžiui, latvių, lietuvių ir lenkų kalbos turi po tris formas ir jos nustatomos skirtingai, anglų kalba turi dvi formas.

Būna programų, kuriose priklausomai nuo situacijos reikia nustatyti būdvardžio, įvardžio, dalyvio giminę (pvz., *Jonas prisijungęs*, *Jonienė atsijungusi*), daiktavardžio (ypač vardo: *Sveiki, Jonai*; *Gautas laiškas iš Jono* ir pan.) linksnį. Linksniavimo ir kitokių žodžių kaitybos uždavinių atskirus atvejus galima rasti programavimo uždavinynuose. Tai lokalizavimo dalykas. Internacionalizuojamoje programoje pakanka numatyti tik galimybę (vietą) tokiems algoritmams.

6 Skaičių formatas

Dešimtainės trupmenos trupmeninė dalis nuo sveikosios skiriama dvejopai: kableliu (Europoje, išskyrus Jungtinę Karalystę, Pietų Afrikos Respublikoje, Pietų Ameri-

koje ir kitur) arba tašku (Jungtinėse Amerikos Valstijose, Jungtinėje Karalystėje, Australijoje, daugelyje Azijos valstybių). Todėl realiųjų skaičių įvedimą ir išvedimą (spausdinimą arba rodymą ekrane) reikia suprogramuoti taip, kad į skaičių būtų įrašomas ženklas, paimtas iš lokalizuojamų išteklių eilutės, turinčios tik vieną ženklą: kablelį arba tašką.

Šį veiksmą reikia programuoti ir bet kurioje kitoje Paskalio kalbos programoje (ne tik internacionalizuojamoje), nes Paskalio kalbos procedūros *read* ir *write* trupmenos skirtuku laiko tik tašką, o lietuvių kalboje trupmenos skirtukas yra kablelis.

Algoritmus galima pateikti funkcijų pavidalu: vieną – įvedamai eilutei keisti realiuoju skaičiumi, kitą – realiajam skaičiui keisti išvedimui skirta eilute.

7 Žodžių rikiavimas

Žodžiai dažniausiai rikiuojami abėcėlės tvarka, tačiau ji įvairiose kalbose nevienoda. Pavyzdžiui, raidė Y lietuvių kalboje eina prieš J, kartais (žodynuose) tapatinama su raide I, o anglų kalbos abėcėlėje užima priešpaskutinę vietą. Rikiavimo tvarkai nustatyti į lokalizuojamuosius išteklius reikėtų įtraukti eilutę, kurioje būtų visos abėcėlės raidės, išdėstytos pagal abėcėlę.

Daugelis kalbų turi dviraidžių ir triraidžių – kai kurios greta parašytos raidės rikiavimo požiūriu laikomos viena raide. Vadinasi, į išteklius reikia įtraukti eilutę su dviraidžių ir triraidžių sąrašu. Čekų lokalizacijos eilutėje būtų vienas dviraidis *ch*, kroatų – trys: *dž lj nj*, vengrų – devyni: *cs dz dzs gy ly ny sz ty zs*, o lietuvių lokalizacijoje ši eilutė būtų tuščia, nes lietuvių kalba dviraidžių neturi (anksčiau buvęs dviraidis *ch* dabar laikomas dviem atskiromis raidėmis). Pradinės programos kūrėjui visų kalbų dviraičių ir triraidžių žinoti nereikia – juos įrašys lokalizuotojai. Tačiau jis turi parašyti rikiavimo programą, kuri deramai panaudotų išteklių eilutėje pateiktą informaciją.

8 Internacionalizavimo klaidų paieška realiose programose

Lietuvių kalba, kaip ir daugelis indoeuropiečių kalbų (čekų, latvių, lenkų, rusų ir kt.), yra sintetinė, turi daug gramatinių formų (daiktavardžiai, būdvardžiai, dalyviai turi linksnius ir gimines, veiksmažodžiai asmenuojami). Anglų kalba analitinė, joje mažiau gramatinių formų. Programoje, parašytoje anglų kalba, panaudojama mažiau gramatinių formų, negu programoje, parašytoje kuria nors sintetine kalba, todėl anglišką programą internacionalizuoti sudėtingiau – reikia papildyti trūkstamomis formomis, nebūdingomis anglų kalbai. Lengva kurią nors pamiršti ir taip padaryti internacionalizavimo klaidą.

Pastebėti internacionalizavimo klaidą pradinėje programoje (t. y. ten, kur iš tikrųjų ji padaryta), sunku arba išvis neįmanoma. Ji išlenda tik lokalizuojant programą ar ją jau lokalizavus.

Turime nemažai lokalizuotų programų. Kone visų jų originalai angliški, todėl klaidų turėtų būti. Iš tikrųjų taip ir yra.

Mokiniai, susipažinę su internacionalizavimo pradmenimis, būtų pajėgūs tokių klaidų ieškoti. Tai būtų geros pratybos ir prasmingas praktinis darbas.

Klaidų ieškoti gali padėti šiam tikslui parengtas klausimynas [3] – užuominos apie labiausiai tikėtinas klaidas.

9 Išvados

Pagrindiniai programų internacionalizavimo elementai gali būti nesunkiai paaiškinti elementarių programų pavyzdžiais. Tai suprastinti pavyzdžiai, gerokai besiskiriantys nuo praktiškai egzistuojančių programų. Tačiau jie visiškai adekvačiai atspindi realių programų principus.

Jeigu besimokantysis turi programavimo pagrindus ir gerus programavimo stiliaus įpročius, t. y. programą rašo ne tik sau, tai jam viso to pakaks, kad ir pats galėtų rašyti internacionalizuotas programas. Tai pasakytina apie Jaunųjų programuotojų mokyklos absolventus arba artėjančius prie jų kitus mokinius, baigusius mokyklinį programavimo modulį ar rimtą programavimo kursą papildomojo ugdymo įstaigoje, jei mokykloje buvo skiriama pakankamai dėmesio programavimo stiliui ir vertinami programų tekstai, o ne vien kompiuterio išduoti rezultatai.

Ieškoti internacionalizavimo klaidų ir sėkmingai jas rasti gali ir silpniau mokantys programavimą – vertinančiam kitų darbą nebūtina jį mokėti atlikti pačiam.

Rastos klaidos padėtų mokiniui pasijusti visaverčiu programinės įrangos projektavimo dalyviu. Internacionalizavimo pradmenys turėtų išplėsti mokinio žinias apie kitas kalbas, padėti pajusti lietuvių kalbos reikšmę ir jos vietą tarp kitų kalbų.

Literatūra

[1] V. Dagienė, G. Grigas, T. Jevsikova. *Programinės įrangos lokalizavimas*. Matematikos ir informatikos institutas, Vilnius, 2010.

[2] B. Esselink. *A Practical Guide to Localization*. John Benjamins Publishing Company, 2000.

[3] G. Grigas. Programinės įrangos testavimas internacionalizavimo požiūriu. *Kalbų studijos*, **17**, 2010.

[4] R. Laucius. Programinės įrangos lokalizavimo kursas. *Informacinės technologijos 2005, Konferencijos pranešimų medžiaga*, pp. 138–141, Kaunas, 2005. Technologija

[5] R. Laucius, V. Dagienė. Lokalizavimo kurso projektavimas. *Lietuvos matematikos rinkinys*, **45**:213–218, 2005.

SUMMARY

Programming style and teaching internationalisation of programms

V. Dagys, G. Grigas

The possibilities to introduce school students to software internationalization element are discussed. The basic knowledge of internalization can be provided in the forma of simple practical tasks of programming. The software internalization learning is a multi-topic: students gain understanding of internationalization and localization of programs, enough knowledge to be able to test actually used programs, especially open source software (this could be a serious motivation), they learn about the world's linguistic diversity, realize deeper meaning of native language and its place among the other languages.

Keywords: teaching programming, software internationalisation, software lokalisation, programming style.