Praktikos darbo programų reikalavimai

Saulius Gražulis

Bioinformatika III

Bioinformatikos kursui reikės parašyti komandinės eilutės programėles *x[1] sistemai. Šios programėlės turi tenkinti tokius reikalavimus:

1. Kiekvienos programos pradžioje turi būti komentaras, vienu dviem sakiniais aprašantis, ką programa daro ir kaip ja naudotis. Taip pat būtina nurodyti, kokius įvesties failus ir kokiais formatais programa skaito, ir kokius išvesties failus bei kokiais formatais užrašo.

2. Programų komentarus galite rašyti angliškai arba lietuviškai. Jei rašote lietuviškai, naudokite lietuviškus rašmenis **UTF-8 koduote**.

   Komentarai turi būti rašomi viena kokia nors kalba (arba tike lietuviškai, arba tik angliškai, bet ne vienoje vietoje vienaip, kitoje kitaip). Galite pateikti ir lietuviškus, ir angliškus komentarus, bet tada visi komentarai turi būti dvikalbiai, ir patartina pažymėti formaliomis, kompiuteriu skaitomomims žymėmis, kokia kalba parašytas komentaras, ir visus komentarus taip apipavidalinti. Pvz.:

   XML formato komentarai:

```
<annotation>
    <documentation lang="en">
        Describe a relational database: inside this element, all
        database tables, their columns (a.k.a fields) and foreign-key
        relations between them are described ....
    <documentation>
    <documentation lang="lt">
        Šiame duomenų elemente pateikiamas reliacinės DB aprašymas ...
    </documentation>
</annotation>
```

   Perl komentarai, pažymėti #+LT, #-LT, #+EN, #-EN žymėmis:

```
#+EN
#* The program reads PDB XML files and outputs their structure IDs,
# DOIs mentioned in thos files and their file names.
#**
#-EN
#+LT
#* Programa skaito PDB XML failus ir išveda failų vardus bei juose
# paminėtus DOI.
#
# PANAUDOJIMAS/USAGE:
#     $0 tests/inputs/PDB/4L*.xml
#     $0 < tests/inputs/PDB/4LDY.xml
#**
#-LT
```

3. Tie patys dokumentavimo reikalavimai galioja kiekvienai funkcijai: komentaruose prieš kiekvieną funkciją reikia vienu ar keletu sakinių aprašyti, kam ši funkcija reikalinga, kokie jos įėjimo parametrai ir kokias ji grąžina reikšmes.

4. Kintamųjų vardai turi būti rašomi viena kalba, arba **tik** angliškai, arba **tik** lietuviškai; kintamųjų vardams būtina naudoti tą kalbą, kurią naudojate

1 Tokios programėlės, beje, veiks ir daugumoje kitų sistemų – DOS, MS Windows, VMS...

komentarams (t. y. jei Jūsų projekte komentarai tik angliški, tai kintamųjų vardams naudokite tik angliškus žodžius ir sanrtumpas).

5. Programos turi skaityti įvesties failus, išvardintus komandinėje eilutėje, ir visus juos paeiliui apdoroti. Jei nei vienas failas nenurodytas, jos turi skaityti duomenis iš savo standartinės įvesties kanalo STDIN.

Perlo kalba šio punkto nuostatą tekstiniams failams realizuoti itin paprasta:

```
#!/usr/bin/perl

# Skaitome visus tekstnius failus, išvardintus komandinėje eilutėje,
# o jei tokių nėra – skaitome STDIN:

use strict;
use warnings;

while( <> ) {
      process( $_ );
}
```

Jei failai binariniai, galima naudoti tokį kodą:

```
#!/usr/bin/perl

# Skaitome visus binarinius failus, išvardintus komandinėje eilutėje, o jei
# tokių nėra – skaitome STDIN:

use strict;
use warnings;

push( @ARGV, "-" ) unless @ARGV; # Skaitysime STDIN ("-") jei nieko nenurodyta

for my $filename (@ARGV) {
      open( INPUT, $filename )
            or die( "can not open file '$filename' - $!" );

      process( \*INPUT );

      my $buffer;
      my $size = 1024;
      read( INPUT, $buffer, $size )
            or die( "error reading file '$filename' - $!");


# .... darome ką reikia ......


      close( INPUT )
            or die( "error closing '$filename' - $!" );
}
```

6. Jei programų rezultatas yra palygint nedidelis tekstinis failas (<100MB), jį reikia išvesti į programos standartinę išvestį, STDOUT. Jei rezultatas binarinis arba labai didelis (pvz., PNG formato paveiksliukai, ODT formato dokumentai, PostScript failai su paveikslėliais ir pan.), reikia pagal nutylėjimą (kitaip tariant, numatytasis elgesys turi būti:) sukurti failą darbinėje direktorijoje, panaudojant programos pavadinimą ir failo formatui tinkamą plėtinį. Pvz., programa, pavaizduojanti baltymo molekulę PNG formato paveikslėlyje, pavadinta `pdb_draw`, turi sukurti darbinėje direktorijoje failą vardu '`pdb_draw.png`', ir į jį užrašyti paveikslėlį. Tačiau tai tik „numatytasis elgesys“ (angl. ”default behaviour”); turi būti galimybė nukreipti rezultatą į failą naudotojo nurodytu vardu.

   Tai galima padaryti keliais būdais:

   1. įgyvendinant programoje neprivalomą argumentą (angl. option) '-o failo-vardas.plėtinys',

2. pasirenkant numatytąjį (default) elgesį tik tada, kai STDOUT yra terminalas; jei STDOUT yra ne terminalas, rašyti binarinį failą į STDOUT, nes tai arba failas, arba kitos programos STDIN.

Tekstinių failų išvedimą į STDOUT Perlo kalboje realizuoti itin lengva, tiesiog reikia panaudoti komandą `print`:

```
print "rezultatas = ", $result, "\n";
```

Binariniai failai Perl'e rašomi funkcijomis `write` arba `syswrite`.

Nustatyti, ar mūsų programos STDOUT yra terminalas, galima taip:

```
Perl kalba:
if( -t STDOUT ) {
      # STDOUT yra terminalas:
      my $output_filename = File::Basename::basename( $0 ) . $extesion;
      close( STDOUT );
      open( STDOUT, ">$output_filename" ) or
            die( "$0: could not reopen STDOUT to '$output_filename': $!" )
}
# Toliau tiesiog rašome print arba write( STDOUT, $buffer ) funkcijomis...
sh ir bash kalbomis:
if [ -t 1 ]; then echo "STDOUT is a TTY" >&2; else echo "not a tty" >&2; fi
C ir C++ kalbomis:
#include <unistd.h>
if( isatty(1) ) {
      /* STDOUT yra terminalas */
}
Kitomis kalbomis – pakviečiant vienaip ar kitaip libc isatty() funkciją... :)
```

Šis metodas nėra labai paplitęs *x terpėje, todėl jį naudojant, reikėtų į STDERR parašyti pranešimą, kad rezultatas užrašomas į failą, nurodant to failo vardą, priežastį, kodėl taip padaryta, ir būdą, kaip nukreipti rezultatą į failą norimu vardu.

Jei naudojama '-o' opcija, galima panaudoti kokį nors Perlo opcijų (komandinės eilutės argumentų) apdorojimo paketą.

Jei rašant programą bus laikomasi nurodytų reikalavimų, taps labai lengva kombinuoti Jūsų parašytas programas su kitomis *x terpėje prieinamomis programomis. Pvz.:

```
# Pasirinkime grandines A, B ir X ir apdorokime jas:
sh> grep -E '^(ATOM  |HETATM).{15}[ABX]' input.pdb | ./my-command > result.pdb
# Suraskime visus PDB failus mūsų medyje ir apdorokime juos:
sh> find pdb/ -name \*.pdb | xargs my-command > result.pdb
```

7. Programos klaidų pranešimai turi būti rašomi į STDERR. Pranešimai turi būti informatyvūs, t.y. juose turi būti nurodomas mažų mažiausiai programos, kurioje įvyko klaida, vardas, failo, kuris buvo apdorojamas, vardas, jei reikia – apdorojamos failo eilutės ir simbolio joje numeriai, ir aiškus bei *mandagus* klaidos aprašymas. Jei sutikti neteisingi duomenys, pacituokite programai netinkančią duomenų reikšmę. Pageidautina, jei įmanoma, taip pat nurodyti, kaip galima būtų pašalinti klaidą sukėlusias aplinkybes, jei tai nėra „kasdieninė Unix naudotojo išmintis“ (t.y. jei tai nėra įprastos failų sistemos situacijos – netinkamos priėjimo teisės,

nesantys failai, perpildyti diskai ir pan.). Jei buvo kviečiama sisteminė funkcija, būtina įtraukti į klaidos pranešimą taip pat ir sistemos diagnostinį pranešimą (Perle jis gaunamas kintamajame $!, C/C++ -- naudojant C bibliotekos kintamąjį/macro 'errno' ir funkcijas strerror() (#include <errno.h>; #include <string.h>)).

Klaidų pranešimai turi būti spausdinami vieningu formatu, tinkamu automatiniam apdorojimui (machine-readable). Tinkami yra GNU (http://www.gnu.org/prep/standards/standards.html), „TurboPascal“ arba „išplėstas Perl“ pranšimų formatai. Šių formatų pavyzdžiai pateikti žemiau:

```
# GNU formatas:

sh> my-command x.pdb y.pdb z.pdb

my-command:x.pdb:20:15: KLAIDA, neteisinga reišmė "ABC" ATOM įrašo X koordinatės lauke
my-command:y.pdb: ERROR, can not open file 'y.pdb' for input – no such file or directory

# „TurboPascal“ formatas:

sh> my-command x.pdb y.pdb z.pdb

my-command:x.pdb(20,15): ERROR, neteisinga reišmė "ABC" ATOM įrašo X koordinatės lauke
my-command:y.pdb: ERROR, can not open file 'y' for input – no such file or directory
my-command:z.pdb: WARNING, PDB chain symbol '1' should be an upper-case letter

# Išplėstas Perlo formatas:

sh> my-command x.pdb y.pdb z.pdb

ERROR, incorrect value "ABC" in the ATOM record X coordinate field of the file 'x.pdb'
at my-command line 8.

sh> perl -e 'open ($ARGV[0]) or die "Could not open file \"$ARGV[0]\" - $!"' xxx.pdb

Could not open file "xxx.pdb" - No such file or directory at -e line 1.
```

Klaidos pranešime rekomenduojama nurodyti klaidos svarbą (lygį): ERROR (KLAIDA) yra situacija, po kurios programa negali tęsti darbo ar failo apdorojimo; po tokios klaidos programa turėtų grąžinti nenulinį statusą. WARNING (PERSPĖJIMAS) pranešimas reiškia, kad įvyko nelaukta situacija, ir nors programa tęsia darbą, rezultatai gali būti neteisingi (programos grįžimo statusas bus arba nulis, arba nenulis, bet skirsis nuo klaidos statuso); galų gale NOTE (PRANEŠIMAS), arba INFO (INFORMACIJA) nurodo, kad viskas vyksta teisingai, bet pateikia reikalingą statistiką ar kitą informaciją apie skaičiavimo procesą (po tokio pranešimo programa žinoma sugrįš su nuliniu statuso kodu). Klaidos pranešime reikėtų vengti skirtukų, naudojamų klaidos pranešimo komponentams atskirti. Pvz. GNU stiliaus pranešimuose pats pranešimo tekstas neturėtų turėti dvitaškio („:“). Taip mes palengvinsime kompiuterinę klaidų analizę. Reikalui esant, pranešimo teksto dalis galima atskirti kableliu, brūkšneliu arba brūkšniu:

```
my-command:y: ERROR, can not open file 'y' for input – no such file or directory

my-command:y: ERROR, can not open file 'y' for input -- no such file or directory

my-command:y: ERROR, can not open file 'y' for input - no such file or directory
```

Nepergyvenkite, kad kartais pranešimų eilutės gaunasi ilgokos – jų **nereikia** dirbtinai „lankstyti“ naujos eilutės simboliais. Niekada nežinote, kokio pločio bus naudotojo ekranas, todėl greičiausiai nepataikysite; „sulankstytas“ eilutes bus sunku apdoroti *x įrankiais (grep, awk); o „sulankstyti“ jas visada galėsime po to komanda 'fold' arba 'less'. Be to, 'less -S' leidžia patogiai peržiūrėti net labai ilgas eilutes, jų nelankstant.

Poetiškos natūros gali pasinaudoti Haiku stiliumi klaidos pranešimui rašyti: (http://www.gnu.org/fun/jokes/error-haiku.html, http://www.salon.com/21st/chal/1998/02/10chal2.html). :)

8. Paprastai klaidų pranešimai spausdinami angliškai. Norint pritaikyti programas (tarp)tautiniei auditorijai, jas reikėtų internacionalizuoti (i17i) ir lokalizuoti (i9i). Teisingiausia tam būtų naudoti specialius paketus, tokius kaip Maketext (http://perldoc.perl.org/Locale/Maketext/Simple.html) arba Gettext (http://linux.maruhn.com/sec/perl-gettext.html). Tačiau tokių sistemų naudojimas nėra paprastas ir užima laiko, todėl nėra jo reikalaujama. Galima naudoti ad-hoc savo sukurtus modulius pranešimų vertimui.

   Jei nenorite užsiimti savo programos lokalizacija, rašykite klaidų pranešimus angliškai. Jei norite rašyti juos lietuviškai, pateikite internacionalizacijos modulį, kuris turėtų lietuviškus ir angliškus pranešimų atitikmenis; programa **turi** spausdinti angliškus, ASCII koduotę naudojančius pranešimus, jei nustatyta lokalė C arba jei nenustatyta jokia. Pranešimai nacionalinėmis kalbomis **turi** būti spausdinami, naudojant UTF8 koduotę; 8 bitų koduotės yra pasenusios ir šiuolaikinei aplinkai netinkamos.

9. Visada naudokite visus kalbos palaikomus klaidų patikrinimo mechanizmus! Perle būtina naudoti 'use strict' ir 'use warnings' pragmas (arba 'perl -w' opciją); C/C++ bei kitose kompiliuojančiose kalbose būtina įjungti visus perspėjimus (pvz. gcc kompiliatoriui naudoti opciją '-Wall'), ir pasiekti, kad visi failai būtų kompiliuojami be perspėjimų.