Ειδικά Θέματα Βιοπληροφορικής

Διάλεξη 3η :

Perl (1) : Γενικά χαρακτηριστικά γλώσσας,
Συγγραφή και εκτέλεση προγραμμάτων, Δομή προγραμμάτων:
το πρώτο πρόγραμμα σε Perl, Τύποι μεταβλητών : scalars,
Οι δύο πρώτες εντολές : for, while, Άσκηση 1η.

Εισαγωγή

Perl είναι τα αρχικά του "Practical Extraction and Report Language". Είναι μία γενική γλώσσα προγραμματισμού που δημιουργήθηκε από τον Larry Wall. Πρόκειται για ανοικτό και ελεύθερο λογισμικό, γεγονός που μαζί με τα ιδιαίτερα χαρακτηριστικά της γλώσσας οδήγησε στην ευρύτατη διάδοση της. Εκδόσεις της perl μπορούν να βρεθούν σχεδόν για το σύνολο των λειτουργικών συστημάτων, από Linux και MacOSX, μέχρι VMS και windoze.

Εισαγωγή

Η perl είναι κυρίως γνωστή για τις πλούσιες δυνατότητες που παρέχει για χειρισμό κειμένου, ιδιαίτερα μέσω των λεγόμενων regular expressions. Δεδομένου ότι η μεγάλη πλειοψηφία των βάσεων δεδομένων βιολογικού περιεχομένου είναι του τύπου flat file, η perl γρήγορα καθιερώθηκε ως μία από τις πλέον ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού για εφαρμογές Βιοπληροφορικής. Επιστέγασμα της στενής σχέσης της perl με τη Βιοπληροφορική είναι η BioPerl, μία έκδοση της perl (στην πραγματικότητα ένα σύνολο από modules) ειδικών για εφαρμογές Βιοπληροφορικής.

Εισαγωγή

Στόχος αυτού του μαθήματος, είναι η εξοικείωση με τα βασικά χαρακτηριστικά της perl, και όχι η χρήση ειδικευμένων εργαλείων όπως η BioPerl. Αυτό όχι μόνο γιατί είναι δύσκολο να χρησιμοποιηθεί η BioPerl απουσία γνώσης της perl, αλλά και γιατί η perl αυτή καθ' αυτή αρκεί την επίλυση μιας πληθώρας προβλημάτων Βιοπληροφορικής.

Πλεονεκτήματα - Μειονεκτήματα

Η perl είναι portable, εύκολη στη χρήση, συμπαγής στο συντακτικό της (ίσως υπερβολικά), ενώ η άνεση της με το χειρισμό αλληλουχιών χαρακτήρων την κάνει ιδανική για εφαρμογές Βιοπληροφορικής.

Από την άλλη μεριά, ίσως να μην είναι η πλέον κατάλληλη γλώσσα για προγράμματα με βαρείς αριθμητικούς υπολογισμούς, ή για προγράμματα μεγάλης έκτασης (για τα οποία μία πιο δομημένη και strongly-typed γλώσσα προγραμματισμού ίσως να είναι καλύτερη).

Σύνταξη προγραμμάτων

Ένα πρόγραμμα σε perl είναι ένα απλό αρχείο κειμένου το οποίο περιέχει τον κώδικα perl.
Για τη σύνταξη, λοιπόν, ενός προγράμματος perl το μόνο που χρειάζεστε είναι ένας κειμενογράφος όπως ο vi, vim, nedit, xedit, joe, emacs, ...

Η συνηθισμένη σύμβαση είναι τα προγράμματα σε perl να έχουν την κατάληξη '.pl' π.χ. myprog.pl

Προστασίες αρχείων perl

Ειδικά για την perl -και για να γίνει η εκτέλεσή τους εύκολη- θα πρέπει να θυμάστε να θέτετε το execute flag για τα αρχεία που περιέχουν κώδικα perl :

```
chmod 755 myprog.pl
```

Εκτέλεση προγραμμάτων

Εάν το αρχείο που περιέχει τον κώδικα έχει το execute flag, και εάν η πρώτη γραμμή αυτού του αρχείου είναι η #!/usr/bin/perl, τότε για την εκτέλεση του προγράμματος αρκεί η αναφορά του ονόματος του αρχείου.

Παρένθεση: compiled ή interpreted ?

Η perl δεν είναι μια κλασσική interpreted γλώσσα: Πριν η εκτέλεση του προγράμματος αρχίσει, ο κώδικας ελέγχεται και μεταγλωττίζεται (εσωτερικά), και αυτό που τελικά εκτελείται είναι το καθ' ολοκληρία μεταγλωττισμένο πρόγραμμα. Αποτέλεσμα αυτού του δυαδισμού, είναι η ύπαρξη ενός μεταγλωττιστή της perl (με το χαρακτηριστικό όνομα 'perlcc').

Το πρώτο πρόγραμμα σε perl

... είναι όπως πάντα βλακώδες, αλλά ...

```
#!/usr/bin/perl -w

print "Hello world!\n";

exit(0);
```

Σύγκριση με τη C

```
#!/usr/bin/perl -w

print "Hello world!\n";

exit(0);
```

- Δεν υπάρχουν τα header files (stdio.h, math.h, ...)
- Δεν δηλώνεται η κυρίως συνάρτηση [main()].
- Το πρόγραμμα εκτελείται με την απλή αναφορά του ονόματος του αρχείου που το περιέχει. Δεν απαιτείται διακριτό στάδιο μεταγλώττισης.
- Η εντολή για την εκτύπωση στην καθιερωμένη έξοδο διαφέρει σημαντικά [print αντί για printf("...")].

Εισαγωγή σχολίων

```
#!/usr/bin/perl -w

# This is a comment :
# The first line above tells the unix shell which
# program to use in order to run this script (in
# this case, the perl executable). For all our
# programs use the line shown above (with the -w
# flag to enable warnings).

print "Hello world!\n";

exit(0);
```

Τύποι δεδομένων

Στη C οι τύποι δεδομένων (όπως int, float, char, ...), είναι καλά καθορισμένοι, η δήλωση τους είναι υποχρεωτική, και οι πιθανές τους χρήσεις περιορίζονται από το χαρακτήρα τους. Στην perl τα πράγματα είναι διαφορετικά. Ο πλέον απλός τύπος δεδομένων της perl είναι τα scalar (βαθμωτές μεταβλητές) των οποίων το όνομα αρχίζει πάντα από το χαρακτήρα $.

Οι βασικότερες ιδιότητες (και διαφορές από την C) των scalars με τη βοήθεια παραδειγμάτων είναι:

Scalars

Η δήλωση των μεταβλητών είναι προαιρετική ...

```
#!/usr/bin/perl -w

$a = 10.0;
$b = 4.0;
print $a+$b;
print "\n";

exit(0);
```

Scalars

... αν και είναι μία καλή συνήθεια να τις δηλώνετε.

```
#!/usr/bin/perl -w

use strict;          # this is a 'perl pragma' that tells
                     # the interpreter to treat usage of
                     # undeclared variables as an error.

my $a;               # Variable declaration using 'my'
my $b;

$a = 10.0;
$b = 4.0;
print $a+$b;
print "\n";
exit(0);
```

Scalars

Οι scalars μπορούν να είναι αριθμοί ή strings ...

```
#!/usr/bin/perl -w

$a = 10.0;
$b = "This is a string";
print $a;
print "\n";
print $b;
print "\n";

exit(0);
```

Scalars

... ή και τα δύο

```
#!/usr/bin/perl -w

$a = 10.0;
print $a;
print "\n";
$a = "This is a string";
print $a;
print "\n";

exit(0);
```

Scalars

Η χρήση τους με την print είναι εύκολη ...

```
#!/usr/bin/perl -w

$a = 10.0;
$b = "This is a string";
print "$a $b\n";

exit(0);
```

Scalars

... αν και υπάρχουν αρκετές παραλλαγές ...

```
#!/usr/bin/perl -w

$a = 10.0;
$b = "This is a string";
print $a, " ", $b, "\n";

exit(0);
```

Scalars & print

... που σε μερικές περιπτώσεις είναι αναγκαίες

```
#!/usr/bin/perl -w

$a = 10.0;
$b = "This is a string";
$c =  4.0;
print "$a+$c $b\n";
print $a+$c, " ", $b, "\n";
print $a+$c, " $b\n";

exit(0);
```

Scalars & print

Προσοχή στη χρήση διπλών και μονών εισαγωγικών

```
#!/usr/bin/perl -w

$a = 10.0;
$b = "This is a string";
$c =  4.0;

print "$a $b $c \$\$\n";
print '$a $b $c\n';
print $a+$c, " $b\n";
print $a+$c, ' $b\n';
print "\n";

exit(0);
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "42";
$b = "58";

print $a+$b, "\n";

exit(0);
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "42";
$b = "58";

print $a+$b, "\n";

exit(0);
```

Παράγει ως έξοδο τον αριθμό 100.

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "a test";

print $a+$b, "\n";

exit(0);
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "a test";

print $a+$b, "\n";

exit(0);
```

Παραδόξως, το πρόγραμμα παράγει ως έξοδο τον αριθμό 0,
αλλά με δύο αναφορές σφαλμάτων :

```
Argument "a test" isn't numeric in addition (+)
Argument "This is " isn't numeric in addition (+)
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "a test";

# The dot below is the perl way to add
# (concatenate) two strings
print $a . $b, "\n";

exit(0);
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "a test";

# The dot below is the perl way to add
# (concatenate) two strings
print $a . $b, "\n";

exit(0);
```

Το πρόγραμμα παράγει ως έξοδο την αλληλουχία χαρακτήρων "This is a test"

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "test number ";
$c = "2";
$d = "32";

print $a . $b . $c, " but $c+$d=", $c+$d, "\n";

exit(0);
```

Scalars

Οι αθόρυβες μετατροπές τύπων της perl ...

```
#!/usr/bin/perl -w

$a = "This is ";
$b = "test number ";
$c = "2";
$d = "32";

print $a . $b . $c, " but $c+$d=", $c+$d, "\n";

exit(0);
```

```
This is test number 2 but 2+32=34
```

Η εντολή for

Η χρήση της for στην perl είναι σχεδόν πανομοιότυπη με αυτή που γνωρίζετε από τη C :

```
#!/usr/bin/perl -w
for ( $i=0 ; $i < 100 ; $i++ )
   {
     print "$i\n";
   }
```

Η εντολή for

```
#!/usr/bin/perl -w

for ( $i=0.0 ; $i < 100.0 ; $i += 2.0 )
  {
    print $i, " ", sqrt( $i ), "\n";
  }
```

Η εντολή for

```
#!/usr/bin/perl -w

for ( $i=0.0 ; $i < 100.0 ; $i += 2.0 )
  {
    print $i, " ", sqrt( $i ), "\n";
  }
```

```
0 0
2 1.41421356237309
4 2
6 2.44948974278318
```

Η εντολή for

Παραλλαγή με printf αντί για print

```
#!/usr/bin/perl -w

for ( $i=0.0 ; $i < 100.0 ; $i += 2.0 )
   {
   printf " %10d %15.5f\n", $i, sqrt( $i );
  }
```

Η εντολή for

Παραλλαγή με printf αντί για print

```
#!/usr/bin/perl -w

for ( $i=0.0 ; $i < 100.0 ; $i += 2.0 )
   {
     printf " %10d %15.5f\n", $i, sqrt( $i );
   }
```

```
         0         0.00000
         2         1.41421
         4         2.00000
....
```

Η εντολή for

Παραλλαγή με printf αντί για print

```
#!/usr/bin/perl -w

for ( $x=0.0 ; $x < 10.0 ; $x += 0.50 )
  {
    for ( $y=0.0 ; $y < 10.0 ; $y += 0.50 )
      {
        printf " %4.2f %4.2f %15.5f\n", $x, $y, sqrt($x) * sin($y);
      }
  }
```

Η εντολή for

Παραλλαγή με printf αντί για print

```
#!/usr/bin/perl -w

for ( $x=0.0 ; $x < 10.0 ; $x += 0.50 )
  {
    for ( $y=0.0 ; $y < 10.0 ; $y += 0.50 )
      {
        printf " %4.2f %4.2f %15.5f\n", $x, $y, sqrt($x) * sin($y);
      }
  }

....
 1.50 5.00        -1.17444
 1.50 5.50        -0.86411
 1.50 6.00        -0.34221
....
```

Η εντολή while

Η χρήση της while στην perl είναι σχεδόν πανομοιότυπη με αυτή που γνωρίζετε από τη C :

```
#!/usr/bin/perl -w
$i = 0.0;
while ( $i < 10.0 )
   {
     print $i * $i, "\n";
     $i++;
   }
```

ΑΣΚΗΣΗ

Δίδεται η συνάρτηση
$\rho = f(x,y) = [\ 10.0 - \mathrm{sqrt}(x^2+y^2)\ ] \cdot \cos[\ \mathrm{sqrt}(x^2+y^2)\ ]$
όπου 'sqrt' είναι η τετραγωνική ρίζα και 'cos'
είναι το συνημίτονο.
Γράψτε ένα πρόγραμμα σε perl το οποίο για κάθε τιμή των x και y, από -10.0 μέχρι και 10.0 και με βήμα 0.10, θα τυπώνει στην καθιερωμένη έξοδο τις τιμές των x, y, και f(x,y) με τη μορφή τριών αριθμητικών στηλών. Στη συνέχεια, χρησιμοποιήστε επανακαθορισμό εξόδου για να σώσετε τα αποτελέ-σματα σε ένα αρχείο με το όνομα results.dat και εξετάστε την γραφική παράσταση της f(x,y) μέσω της χρήσης του προγράμματος gnuplot ως εξής :

ΑΣΚΗΣΗ

```
# gnuplot
...
gnuplot> splot "results.dat" with dots
```

Με βάση αυτή τη γραφική παράσταση, βρείτε ποιο είναι το ολικό μέγιστο της συνάρτησης, και σε ποιες τιμές των x και y αντιστοιχεί.

Στείλτε το πρόγραμμα που γράψετε στο διδάσκοντα εισάγοντας ως σχόλιο (στον κώδικα του προγράμματος) τις απαντήσεις σας :

```
mail -s "My name, Pract 1" glykos@aspera.cluster.mbg.gr < myprog.pl
```