Corso di Laurea di Primo Livello
Scuola Universitaria Interfacoltà
in Biotecnologie
Università degli Studi di Torino

Università di Torino
MBC
Molecular Biotechnology Center

Corso di Abilità Informatiche
Secondo Modulo

AA 2008/2009

LABORATORIO INFORMATICO, LEZIONE 10:

PROGRAMMAZIONE IN PERL – PRIMI PASSI

PREAMBOLO

Finora abbiamo imparato ad usare in modo flessibile programmi già esistenti, ma questo non basta sempre a risolvere i nostri problemi. A volte non esiste (o non conosciamo) il programma che fa per noi anche se i compiti che dovrebbe svolgere non sono molto complicati. In questo caso può essere utilissimo sapere scrivere dei programmi.

Scaricate il file `AbInfo-II_lab10-esercizi.zip` dal sito del corso che, scompattandolo con `unzip` all'interno della directory `home/studente/`, crea la cartella `lab10-perl` in cui troverete alcuni file e che potete usare anche nei prossimi laboratori. **Vi consigliamo di copiarvi dopo ogni laboratorio i programmi scritti da voi su una chiavetta USB o di mandarli ad un vostro indirizzo email per non perdere il lavoro svolto in precedenza!**

ESERCIZIO 1: Lanciare dei programmi in Perl

Il primo passo importante è saper lanciare dei programmi scritti in Perl, non solo per poterli eseguire quando sono pronti, ma anche per controllare e testare il loro corretto funzionamento quando li state scrivendo.

1. Nella cartella `lab10-perl` troverete il file `lanciami.pl` che contiene un programma. Lanciatelo prima usando il suo nome (identificativo) come argomento sulla command line del programma `perl` (l'interprete che esegue le istruzioni dei programmi scritti in Perl).
2. Provate a lanciare il programm direttamente dandogli prima i permessi di esecuzione (ricordate come si fa?) e poi scrivendo:
   ```
   ./lanciami.pl
   ```
   Nota: Il punto all'inizio del comando indica che il programma si trova nella cartella in cui vi trovate attualmente. La shell non riesce a trovarlo altrimenti perché quella cartella non fa parte di uno dei path di esecuzione predefiniti (al contrario del programma `perl` per esempio che si trova su uno dei path standard).
3. Come fa la shell a sapere che il programma dev'essere eseguita dall'interprete di Perl se lo lanciate come nel punto 2? Il programma è un semplice file di testo, guardate la sua prima riga,

cosa notate? Impareremo più in là cosa significa, ma forse avete già un'idea.
4. Guardate brevemente la pagina `man` di `perl`. Potrebbe esservi utile in futuro.

ESERCIZIO 2: Messaggi di errore

Vedremo un attimino alcuni messaggi di errore che restituisce l'interprete `perl` se qualcosa non va nel programma. Non impareremo ancora il significato preciso di tutti questi messaggi, ma è comunque importantissimo prendere familiarità con questi messaggi perché sono spesso utilissimi per sapere cosa (o almeno dove) bisogna correggere se qualcosa non funziona.

In questo esercizio usate l'interprete `perl` sempre con l'opzione `-w` che stampa oltre a messaggi di errore degli utili messaggi di "warning" che spesso indicano problemi anche se non sono errori grave che impediscono l'esecuzione del programma.

1. Provate a eseguire di nuovo il programma del primo esercizio, questa volta pero facendovi informare di eventuali warning.
   Cosa notate? C'e' un problema che non impedisce l'esecuzione del programma che però potrebbe portare a comportamenti inaspettati (anche se in questo caso non lo fa). Per adesso non ci interessa la natura esatta del problema, ma vi mostra che usando l'opzione `-w` potete avere informazioni su eventuali problemi di cui altrimenti non ve ne accorgete. Quindi: usatela sempre quando scrivete i vostri programmi Perl!
2. Nella cartella `lab10-perl` si trovano anche i programmi `miRNA.pl` e sette programmi derivati che contengono piccoli errori. Lanciate prima il programma `miRNA.pl`, guardate l'output che viene stamapto sulla shell e leggete il suo codice sorgente (l'insieme di istrizioni del programma). Poi lanciate uno dopo l'altro i programmi che contengono gli errori. Leggete bene il messaggio di errore (se c'è) e se possibile provate a trovare l'errore nel codice sorgente e correggerlo. Chiedete se avete dei dubbi!

ESERCIZIO 3: Variabili e print

Dopo aver visto alcuni dei possibili messaggi di errore che si possono ricevere, iniziamo a scrivere alcuni programmi semplici.

1. Il primo programma in assoluto che si impara a scrivere in ogni linguaggio di programmazione è l'ormai famoso programma **"Hello, world!"**, cioè un programma che non fa altro che "salutare il mondo" scrivendo esattamente quella stringa come output. Aprite un editore di testo (`gedit`). Scrivete nella prima riga la *command interpretation*, che indica che il programma dev'essere eseguito dall'interprete perl (usate anche l'opzione `-w` per avere eventuali warning). Poi scrivete l'istruzione che scrive "Hello, world!" sullo schermo. Salvate il programma con un nome arbitrario (con estensione "`.pl`"), dategli i permessi di esecuzione e lanciatelo direttamente (non come argomento sulla command line dell'interprete `perl`).
2. Scrivete un programma che concatena due stringhe (sequenze, nomi, ... a voi la scelta, ad esempio "Sono " e "biologo/a") nei 3 modi possibili e stampa per controllo ogni volta la stringa concatenata sullo schermo. Importante: imparate a commentare bene i vostri programmi, quindi aggiungete dei commenti che descrivono cosa fanno le istruzioni (i commenti, per convenzioni, spesso si scrivono nella riga precedente all'istruzione, oppure sulla stessa riga *dopo* la fine dell'istruzione che viene determinata dal ";")
3. Scrivete un programma che dichiara due variabili e le inizializza a valori (o stringhe) arbitrari (ad esempio "ACGT" e "GTTTACCT"). Poi usa una terza variabile e le assegna prima il valore della prima variabile, lo stampa sull'output e poi le assegna il valore della seconda variabile (e lo stampa sull'output).
4. Scrivete un programma che dichirara una variabile e la inizializza a un valore arbitrario (es. "Homo sapiens"). Poi dichiara una seconda variabile e le assegna il valore della prima. Poi

cambia il valore della prima variabile settandolo ad un valore diverso (es. “Mus musculus”). Cosa succede con la seconda variabile che era uguale alla prima? Cambia valore anche questa? Come fate a scoprirlo?

5. Scrivete un programma con due variabili `$var1` e `$var2` che abbiano i valori “`TATA`” e “`CGCG`”. Scrivete una serie di istruzioni per scambiare i valori tra le due variabili (cioè `$var1` deve alla fine avere il valore che inizialmente aveva `$var2` e vice versa). Controllate ogni passo facendovi scrivere il valore dei variabili sullo schermo. Commentate bene il programma.

Alla fine di tutti gli esercizi, ricordatevi di cancellare la cartella `lab10-perl` con tutto il suo contenuto. Se volete salvate i vostri file prima su una chiavetta USB o mandateveli via email.