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

Corso di Abilità Informatiche
Secondo Modulo

AA 2008/2009

LABORATORIO INFORMATICO, LEZIONE 13:

PROGRAMMAZIONE IN PERL – ARRAY E IF-THEN(-ELSE)

ESERCIZIO 1: If-then(-else)

Le istruzioni condizionali sono importantissime per la programmazione in (quasi) tutti i linguaggi di programmazione perché aumentano notevolmente la flessibilità di un programma. Spesso è necessario eseguire certe istruzioni solo se si sono verificate determinate condizioni.

1. Scrivete un programma che apre un file di nome `species.txt` in lettura e legge la sua prima riga. Se la prima riga corrisponde alla specie “Homo sapiens” il programma scrive “uomo” sulla shell. Dopo scrive “ho letto il file”, indipendentemente da cosa ha trovato nella prima riga. Create il file `species.txt` (una volta con la specie giusta, una volta con un'altra specie) e testate il programma.
2. Modificate il programma per farlo scrivere “non si tratta della specie umana” nel caso la prima riga di `species.txt` non fosse “Homo sapiens”. Testate il programma.
3. Scrivete un programma che legge la prima riga di un file di nome `genes.txt`. Se la prima riga corrisponde a “BRCA1” o “BRCA2” il programma scrive “uno dei geni coinvolti in breast cancer”, altrimenti scrive “gene sconosciuto”.
4. Scrivete un programma che legge la prima riga di un file di nome `number.txt`. Se la prima riga è un numero intero tra 5 e 10 il progamma scrive “tra 5 e 10”, altrimenti scrive “non tra 5 e 10”.
5. Scrivete un programma che legge la prima riga di un file di nome `number.txt`. Se la prima riga corrisponde al numero 11 il programma scrive “ok”, altrimenti scrive “non ok”.
6. Modificate il programma precedente *cambiando un solo carattere* per invertire il suo comportamento.

ESERCIZIO 2: If-then-elsif(-else) e altro

Spesso esistono più di due condizioni diversi che richiedono istruzioni specifiche. Potrebbe essere per esempio importante trattare una sequenza in modo diverso secondo il suo tipo (DNA genomico, cDNA, mRNA, miRNA, ...).

1. Scrivete un programma che apre un file di nome `species.txt` in lettura e legge la sua prima

riga. Se la prima riga corrisponde alla specie “Homo sapiens” il programma scrive “uomo” sulla shell. Se invece corrisponde a “Mus musculus” scrive “topo” sulla shell, se la riga corisponde a “Drosophila melanogaster” scrive “mosca”. Altrimenti scrive “specie sconosciuta”.

2. Scrivete un programma che legge la prima riga di un file di nome `number.txt`. Se la prima riga è un numero intero superiore a 10, allora il programma controlla anche la seconda riga del file: se il numero della seconda riga è inferiore a 20, allora il programma scrive “prima riga superiore a 10 e seconda inferiore a 20” altrimenti scrive “prima riga superiore a 10, ma seconda non inferiore a 20”.

ESERCIZIO 3: Array

Uno delle più importanti strutture dati (data structures) è l'array che può essere immaginato come un vettore (in senso matematico, non biologico!) di elementi (normalmente) dello stesso tipo: un array di numeri interi, un array di stringhe, ecc. (in Perl è anche possibile avere diversi tipi di elementi nello stesso array, ma viene fatto raramente e generalmente è da sconsigliare).

1. Scrivete un programma che crea un array contenente le stringhe “genomic DNA”, “cDNA”, “RNA” e “mRNA”. Fate scrivere al programma gli elementi del array sullo schermo, un elemento per riga. Fate attenzione come specificate il ***primo*** elemento dell'array! Sapete perché viene specificato così?
2. Come 1. ma fate scrivere al programma gli elementi sulla stessa riga senza specificare i singoli elementi (specificando l'intero array). Ci sono due modi diversi, quali? Qual'è la differenza? (Guardate le slides della lezione).
3. Come 1. Fatevi scrivere gli elementi dell'array sullo schermo (nel modo che preferite) e fatevi stampare anche la lunghezza dell'array (il numero di elementi). Poi aggiuntete un nuovo elemento “protein sequence” prima all'inizio dell'array, poi alla sua fine. Fatevi di nuovo stampare gli elementi (e la lunghezza) sullo schermo per controllare l'array modificato.
4. Come 3. Questa volta togliete il primo elemento salvandolo in una variabile e fatevi stampare sia il contenuto della variabile (che dovrebbe corrispondere all'elemento eliminato) che l'array modificato e la sua lunghezza.
5. Come 4., ma eliminate l'ultimo elemento invece del primo.
6. Scrivete un programma che crea un array contenente i numeri naturali da 1 a 5. Fatevi scrivere il contenuto dell'array sullo schermo. Poi invertite l'array per avere elementi decrescenti da 5 a 1. Stampate l'array per controllarlo. Alla fine calcolate sia la somma che il prodotto di tutti gli elementi e fatevi stampare i risultati.
7. Come 1. Fatevi scrivere l'array, poi sostituite il primo elemento con la traduzione in italiano “DNA genomico”. Fatevi scrivere il risultato. Provate a fare la sostituzione con diversi metodi:
    - usando l'espressione che definisce l'elemento specifico (`$array[N]`).
    - usando il comando `splice`.
    - cancellando il vecchio l'elemento (`shift`) per poi inserire quello nuovo (`unshift`).