

# Esame di Laboratorio di Fisica Computazionale

8 novembre 2016, ore 9.30

## shell scripting

Si utilizzi `sed` per modificare il file `modello.txt`

- 1) cancellando le prime tre righe che contengono degli asterischi,
- 2) sostituendo le scritte generiche `cognome` e `nome` con il vostro nome e cognome. Si salvi il risultato in un nuovo file.

## Mathematica

1. Si definisca, senza utilizzare la notazione in componenti, una funzione che riceve in input due vettori tridimensionali e che restituisce in output l'angolo da essi formato.
2. Sia data la seguente terna di vettori tridimensionali

$$\vec{v}_1 = (3, 5, 1), \quad \vec{v}_2 = (1, -2, 1), \quad \vec{v}_3 = (-3, 5, 7). \quad (1)$$

Si calcoli, sempre senza utilizzare la notazione in componenti, una terna ortonormale  $\vec{w}_i$  formata dal vettore  $\vec{w}_1 = \vec{v}_1$  opportunamente normalizzato e da altri due versori  $\vec{w}_2$  e  $\vec{w}_3$ .

3. Si consideri la base cartesiana

$$\vec{e}_1 = (1, 0, 0), \quad \vec{e}_2 = (0, 1, 0), \quad \vec{e}_3 = (0, 0, 1). \quad (2)$$

e si scriva l'equazione della retta intersezione dei piani  $(\vec{e}_1, \vec{e}_2)$  e  $(\vec{w}_1, \vec{w}_2)$ .

Si disegnino in un grafico 3D simultaneamente i due piani.

4. La retta del punto precedente prende il nome di asse dei nodi. Si calcolino le componenti del versore  $\vec{n}$  associato a questa retta
5. Si calcoli l'angolo  $\alpha$  formato dal versore  $\vec{n}$  con il versore  $\vec{e}_1$ .  
Si calcoli l'angolo  $\beta$  compreso tra i versori  $\vec{e}_3$  e  $\vec{w}_3$ .  
Si calcoli l'angolo  $\gamma$  compreso tra i versori  $\vec{n}$  e  $\vec{w}_1$ .
6. Si definiscano due funzioni  $R_x(\theta)$  e  $R_z(\theta)$  che restituiscono le matrici di rotazione di un vettore tridimensionale che viene ruotato di un angolo  $\theta$  rispettivamente attorno all'asse  $x$  e  $z$  del sistema di riferimento corrente.
7. Gli angoli  $\alpha$ ,  $\beta$  e  $\gamma$  sono gli angoli di Eulero per il cambio di coordinate dal sistema di riferimento  $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$  a quello  $(\vec{w}_1, \vec{w}_2, \vec{w}_3)$ . A questo scopo si compongano le tre rotazioni nel seguente ordine: prima una rotazione di  $\alpha$  attorno all'asse  $z$ , poi una rotazione di  $\beta$  attorno all'asse  $x$  e infine una rotazione di  $\gamma$  di nuovo attorno all'asse  $z$ .

# C++

Si risolva l'esercizio proposto. Per facilitare la correzione, se possibile includere tutto in un unico file sorgente. La sufficienza è raggiunta risolvendo correttamente i primi quattro punti.

## Esercizio

1. Si scriva una classe `Vector2`, che rappresenterà un vettore in due dimensioni. Si mettano tra i membri *private* le due coordinate reali. Si scriva un opportuno costruttore che inizializzi il vettore, con *valore di default* impostato al vettore nullo.
2. Si scriva il costruttore di copie.
3. Si scriva una funzione membro pubblica `dot` che restituisca il prodotto scalare tra due vettori, e una funzione membro pubblica `norm` che restituisca la norma del vettore.
4. Si scriva una funzione membro pubblica `normalized` che restituisca il vettore normalizzato (senza modificare il vettore stesso). Questa funzione dovrà essere definita virtuale per risolvere i punti successivi.
5. Si scriva una funzione globale `angle` che prenda due `Vector2` e restituisca l'angolo compreso, in gradi. Si valuti se passare i parametri per copia o per referenza, volendo ottenere un comportamento polimorfico per eventuali classi che ereditino da `Vector2` (come al punto 7).
6. Nel `main` si istanzino i due vettori  $(2, 2)$  e  $(1, -1)$  e si verifichi che `v.dot(w)` e `w.dot(v)` danno lo stesso risultato. Si verifichi inoltre che l'angolo compreso tra loro è retto.
7. Si scriva una funzione `FastVector2` che erediti pubblicamente da `Vector2`. Questa classe sopprimerà che il vettore sia normalizzato fin dalla sua costruzione, ma senza mai controllare che questo vincolo sia soddisfatto: lo scopo è quello di ridurre il numero di calcoli eseguiti. Si scriva un opportuno costruttore e un opportuno override della funzione `normalized`. Nel `main` si verifichi che i due versori  $(1, 1)/\sqrt{2}$  e  $(1, -1)/\sqrt{2}$  definiscono un angolo retto.
8. Rispondere brevemente a questa domanda: cosa cambierebbe se la funzione `normalized` non fosse virtuale?
9. Nel `main`, riempire un `std::vector` di 10 000 `Vector2`, con coordinate random nel primo quadrante (può essere utile usare `drand48`, che genera numeri pseudo-casuali tra 0 e 1). Ordinare il vettore con l'algoritmo `std::sort(it1, it2, pred)`, (che ordina gli elementi di un container tra gli iteratori `it1` compreso e `it2` escluso, usando per il confronto il predicato binario `pred`). Il comportamento deve essere quello di ordinare per crescenti angoli, misurati relativamente al versore  $e = (1, 0)$ . Controllare infine che il primo elemento formi con  $e$  un angolo vicino a 0 e l'ultimo un angolo vicino a  $90^\circ$ .

10. Scrivere un template di funzione `genericPred` che realizzi il predicato del punto precedente in versione generica, cioè parametrizzando il tipo dei due argomenti in ingresso. Eseguire quindi l'algoritmo `sort` come nel punto precedente ma usando `genericPred`.