Laboratorio di Informatica per la Fisica II
Appello di febbraio 2013

Indice

Sommario

Scegliere e svolgere, fra i testi proposti, uno o più esercizi in modo tale da raggiungere **5 punti**. I programmi dovranno essere compilati, eseguiti e discussi al momento dell'esame. Alla prova d'esame è necessario presentarsi con una copia funzionante dei programmi in modo da poterli testare sui computer del laboratorio o su quelli dei docenti.

1 Implementazione dell'esperimento della Tavola di Galton - 3 punti + 1 punto

Immaginate una tavola piana, liscia, leggermente iclinata rispetto all'orizzontale.

In essa sono piantati dei pioli, allineati per righe orizzontali (N pioli per riga e N righe).

Sia 1 la spaziatura tra i pioli e la distanza tra le righe, e i pioli siano piantati in modo che i pioli della riga k+1 siano allineati col centro degli spazi della riga k.

Si lasci ora scorrere dal centro del lato rialzato della tavola una sfera omogenea, di diametro appena sufficiente a passare tra due pioli; ad ogni urto la sfera non potrà che spostarsi con uguale probabilità p=0.5 o a destra o a sinistra di una distanza pari a 0.5, per cadere sul piolo successivo. Il tutto avviene N volte, e alla fine, se rilasciamo in sequenza M sfere, queste si distribuiranno, a seconda degli spostamenti subiti, in appositi contenitori allineati con le aperture dell'ultima fila di pioli.

a) si simuli il processo di N urti successivi per M tentativi e si produca la distribuzione di frequenza risultante in funzione del numero (etichetta) della cella.

b) si fitti la distribuzione ottenuta con una Gaussiana

c) **(+ 1 punto)** si sovrapponga il grafico della distribuzione binomiale attesa, dati p e N

d) si confronti la varianza ottenuta dal fit Gaussiano con quella sperimentale e quella attesa dalla distribuzione c

e) si confrontino i valori di chi-2 ottenuti nei casi b e c.

2 Fit di un picco con funzione di Lorentz e di Gauss e confronto dei risultati statistici - 3 punti

Viene fornito un file di testo `Gau-Lor.txt` contenente una sequenza di N dati in formato testo, rappresentanti l'esito di una serie di misure con coppie di valori Energia e relativa frequenza.

Si chiede di:

a) rappresentare in istogramma i dati.

b) fittare i dati con un fondo parabolico cui è sovrapposta una Gaussiana

c) fittare i dati con un fondo parabolico cui è sovrapposta una Lorentziana

Confrontare i risultati b e c usando i parametri statistici noti.

3 Produrre un programma che calcoli il valore di π con il metodo dell'ago di Buffon - 3 punti

Una delle prime applicazioni del metodo Monte Carlo risale al 1777 e consente di stimare il valore di π, con un esperimento di tipo *Hit or Miss*, noto con il nome di *Ago di Buffon*.

Nell'esperimento originale, vengono tracciate sul pavimento delle linee parallele a distanza d l'una dall'altra; viene poi lasciato cadere un ago di lunghezza d assumendo che la sua distanza dalla linea appena inferiore (*pos*) e l'angolo θ, che esso forma rispetto alla perpendicolare delle linee, siano variabili casuali uniformemente distribuite (cfr Fig. 1).

Se l'ago incrocia una linea, allora si avrà un risultato di tipo *Hit*, altrimenti si avrà un *Miss*.

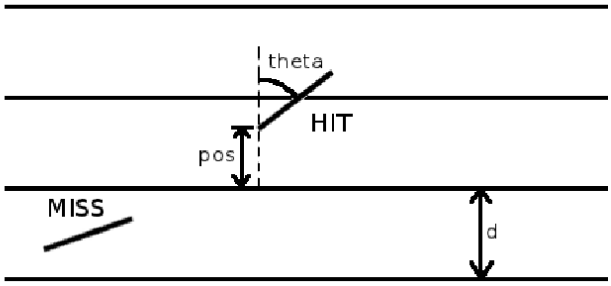

Figura 1

Per un dato ago θ, la probabilità condizionata di un risultato di tipo *Hit*, è data dal rapporto tra la proiezione dell'angolo sulla perpendicolare alle linee e la distanza tra le linee:

$$p\left(Hit|\theta\right) = \frac{d\cos\theta}{d} = \cos\theta$$

Di conseguenza, la probabilità di ottenere un *Hit*, indipendentemente da θ è pari a:

$$p\left(Hit\right) = \int_0^{\pi/2} p\left(Hit|\theta\right) p\left(\theta\right) d\theta = \int_0^{\pi/2} \cos\theta \frac{1}{\left(\frac{\pi}{2} - 0\right)} d\theta = \frac{2}{\pi}$$

- Scrivere un programma che stimi il valore di π simulando l'esperimento dell'ago di Buffon. Si utilizzi la funzione rand() per generare numeri casuali uniformemente distribuiti.
- Si ripeta M volte l'esperimento del lancio di N aghi in modo da ad accumulare una buona statistica dei risultati di π in un istogramma di root. In particolare, provare fissando N pari a 100, 500, 1000, 5000 e 10000.

Fittare tutti gli istogrammi ottenuti con una funzione gaussiana e verificare che l'incertezza della stima di π varia come $1/\sqrt{N}$, calcolando il valore di $\sigma \times \sqrt{N}$.

Il cosiddetto metodo *crude Monte Carlo* consente di stimare l'integrale di una funzione $y(x)$ estraendo numeri casuali x_i uniformemente distribuiti nell'intervallo di integrazione $[a, b]$ e calcolando il prodotto tra il valor medio dei valori y_i corrispondenti e l'ampiezza dell'intervallo (cfr. Metzger 6.2).

- Si applichi il metodo *crude Monte Carlo* per stimare l'integrale di $p(Hit)$ e calcolare il valore di π variando il numero di estrazioni N come nel caso precedente (N=100, 500, 1000, 5000 e 10000). Calcolare di nuovo il valore di $\sigma \times \sqrt{N}$. Quale dei due metodi risulta più accurato? *Hit or Miss* o *crude Monte Carlo*?

4 Creare un programma che esegua cifratura e decifratura di un messaggio - 3 punti

Definire una classe astratta `messaggio` in grado di contenere un messaggio testuale (rappresentato da un oggetto `std::string`) e provvista di due metodi virtuali:

- `virtual void messaggio::cifra(int k)`
  Metodo che effettua la cifratura del messaggio utilizzando il numero intero `k` come chiave di cifratura.

- `virtual void messaggio::decifra(int k)`
  Metodo che effettua ula decifratura del messaggio utilizzando il numero intero `k` come chiave di cifratura.

Definire e implementare, quindi, almeno una sottoclasse tra le due proposte di seguito. Le due classi proposte realizzano due meccanismi di cifratura/decifratura differenti:

- `class rotate : public messaggio`
  Questa classe usa un metodo di cifratura che sostituisce ogni lettera della stringa di messaggio con quella che la segue di `k` (valore della chiave) posizioni nell'insieme delle codifiche ASCII. La decifratura avviene in modo simile: ogni lettera del messaggio cifrato viene sostitita con la lettera che la precede di n posizioni nell'insieme delle codifiche ASCII.

- `class exclor : public messaggio`
  Questa chiave usa un metodo di cifratura che esegue, carattere per carattere, lo xor tra la stringa del messaggio e il valore `k` della chiave. La decifratura avviene eseguendo esattamente la stessa procedura sul testo cifrato.

Creare quindi un semplice `main()` che chieda all'utente di inserire un messaggio da tastiera e poi lo stampi a video dopo averlo cifrato e decifrato.

5 Creare un programma che generi delle matrici quadrato magico - 4 punti + 1 punto

Scrivere un programma che verifichi se una matrice di numeri interi sia un quadrato magico. Una matrice è detta quadrato magico se:

1. la matrice è quadrata (n*n) e contiene tutti i numeri da 1 a n*n senza ripetizioni;
2. la somma degli elementi di ogni riga, di ogni colonna e delle diagonali è identica

Un esempio di quadrato magico 3x3 è:

2	9	4
7	5	3
6	1	8

Il programma deve definire una classe `quadrato_magico` atta a descrivere matrici quadrate. La classe deve avere i seguenti metodi e costruttori:

- `quadrato_magico::quadrato_magico(int s)`
  Questo costruttore crea una matrice quadrata di s*s elemeni. Il costruttore deve inoltre valorizzare le varie righe e colonne in modo casuale con i numeri interi compresi tra 1 e s*s.
- `quadrato_magico::~quadrato_magico()`
  Il destructor deve liberare correttamente la memoria associata alla matrice.
- `void quadrato_magico::stampa() const`
  Il metodo stampa deve mostarare su console una stampa della matrice mostrando tutti gli elementi intabellati per righe e per colonne.
- `bool quadrato_magico::verifica() const`
  Il metodo verifica se la matrice è un quadrato magico verificando le due condizioni sopra descritte. Se la matrice è un quadrato magico il metodo restituirà `true`, altrimenti `false`.

Scrivere anche un semplice `main()` che:

- utilizzi un metodo try-and-catch per generare un quadrato magico n*n con un valore n letto da tastiera;
- stampi a video il quadrato magico trovato (ed eventualmente il numero di matrici casuali che sono state create prima che venisse generato un quadrato magico);
- **(+ 1 punto)** calcoli, con un metodo Montecarlo, la probabilità che una matrice 3*3 creata in modo casuale dalla nostra classe sia un quadrato magico.

6 Programma che simuli un sistema di tre corpi in gravitazione - 5 punti

Dati tre corpi celesti di masse m_1, m_2, m_3, posizioni iniziali (vettori nel piano dei 3 corpi) r_1, r_2, r_3 e velocità iniziali (vettori nel piano) v_1, v_2, v_3, calcolare, usando la legge di gravitazione di Newton (vettoriale), le orbite dei tre corpi e rappresentarle nel piano. Per l'integrazione delle equazioni di moto si usino le espressioni (vettoriali):

$$v_1 = a\Delta t + v_0$$

$$r_1 = v_0\Delta t + r_0$$

Al termine di ogni ciclo (step temporale Δt) si aggiornino le posizioni e velocità iniziali con quelle appena calcolate.

Si suggerisce di partire dal calcolo di un sistema noto (Sole, Terra, Marte), per cui è ragionevole usare uno step $\Delta t = 1$ giorno.

È inoltre opportuno gestire i calcoli usando due classi:

a) una classe vettore in 2D che memorizzi le coordinate x e y dei vettori, e che consenta di fare su di essi tutti i calcoli necessari (+, -, =, mod(), etc.)

b) una classe Pianeta che memorizzi massa, posizione, velocità del corpo celeste, e che consenta di fare con esse i calcoli necessari (accelerazione, distanze relative, determinazione della posizione, stampa dati, etc.)

7 Creare un programma che simuli un libretto esami - 2 punti

Creare una classe `libretto` che possa essere utilizzata per simulare un libretto universitario.

La classe deve avere i seguenti metodi e costruttori:

- `libretto::libretto(std::string nomefile)`
  Questo costruttore crea un oggetto libretto leggendone i parametri da un file di testo che abbia nome `nomefile`. Il file dovrà contenere una prima riga con tre campi: nome dello studente, cognome dello studente, numero di matricola dello studente. Dopo di che il file conterrà tante righe quanti sono gli esami nel piano di studi dello studente. Ogni riga esame conterrà due valori: il nome dell'esame e il numero di crediti dell'esame. Il costruttore deve quindi leggere tutti gli esami e inserirli in un `std::map` che abbia come chiave il nome dell'esame e come valore un array di `float`. Il primo conterrà il numero di crediti dell'esame e il secondo verrà inizializzato a 0 e conterrà il voto ottenuto all'esame dallo studente. Un esempio di file di input è fornito nel file: `libretto.txt`.

- `libretto::libretto(const libretto& l)`
  Copy constructor.

- `libretto::~libretto()`
  Il destructor deve liberare correttamente la memoria associata al libretto.

- `void libretto::stampa() const`
  Il metodo stampa deve mostarare su console una stampa del libretto mostrando le informazioni dello studente e una tabella con tutti gli esami, il numero crediti per ogni esame e l'eventuale votazione (se l'esame è già stato sostenuto).

- `float libretto::media() const`
  Il metodo deve restituire una media di tutti i voti di esame, considerando i soli esami già sostenuti.

- `float libretto::media_ponderata() const`
  Il metodo deve restituire una media ponderata (quindi pesata con i crediti relativi per ogni esame) di tutti i voti di esame, considerando i soli esami già sostenuti.

- `int libretto::voto_laurea() const`
  Il metodo deve restituire il voto di partenza per la laurea. Questo metodo sfrutterà il metodo `media_ponderata` e normalizzerà il risultato in base 110.

- `bool libretto::sostieni_esame(std::string esame)`
  Il metodo dorvrà simulare il sostenimento di un esame. Per far questo genererà un numero casuale tra 1 e 30. Se il numero generato sarà minore di 18, allora l'esame non sarà superato e il metodo restituirà `false`. Altrimenti aggiornerà la mappa degli esami, aggiungendo la votazione dell'esame sostenuto con successo e restituirà `true`.

Scrivere anche un semplice `main()` che crei un oggetto `libretto` e testi le funzionalità implementate.