SAPIENZA
UNIVERSITÀ DI ROMA

Corso di Laurea in Fisica
Laboratorio di Calcolo - **Esercitazione n. 7**

Attorno alla Terra orbitano ormai diverse centinaia di migliaia di detriti derivanti dal danneggiamento di satelliti e altri oggetti artificiali orbitanti. Devi scrivere un programma che simula il moto di questi detriti, distribuiti in modo casuale su un'orbita a 400 km dalla Terra, il cui raggio è di 6 136 km. Il raggio dell'orbita di ciascuno di questi detriti è distribuito come una Gaussiana di ampiezza pari a 150 km.

In questa esercitazione devi scrivere due programmi per simulare il moto di spazzatura spaziale: uno che genera la posizione dei detriti secondo quanto illustrato sopra e uno che permette di osservare il moto animato di questi oggetti attraverso l'uso di `gnuplot`. I detriti sono distribuiti uniformemente lungo l'orbita con una separazione angolare media $\Delta\theta$ e deviazione standard trascurabile. Segui la traccia che segue. Verifica periodicamente che il programma si comporti come atteso.

(1) Definisci una funzione `readFloat()` che richiede l'inserimento di un numero reale positivo all'utente e lo restituisce. Qualora l'utente immetta caratteri non ammessi, la funzione deve reiterare la richiesta.
(2) Attraverso questa funzione ottieni il valore di $\Delta\theta$ dall'utente.
(3) Chiedi all'utente di inserire il nome del *file* su cui scrivere i dati dei detriti e apri il *file* per la scrittura in binario.
(4) Partendo da $\theta = 0$, a passi di $\Delta\theta$, genera casualmente le coordinate $(x, y)$ dei detriti fino a coprire tutto l'angolo giro. I detriti dovranno trovarsi a una quota $h$ variabile dalla superficie terrestre. In media $h$ vale $\langle h\rangle = 400$ km, con una deviazione standard $\sigma \simeq 150$ km. Per generare la quota si può usare la trasformazione di Box–Müller, secondo la quale, se $u_1$ e $u_2$ sono due numeri casuali uniformemente distribuiti nell'intervallo $[0, 1]$, il numero $z = \sqrt{-2\log u_1}\cos(2\pi u_2)$ è un numero casuale distribuito come previsto dalla distribuzione di Gauss. $z$ ha valor medio nullo e deviazione standard unitaria, perciò un numero con deviazione standard $\sigma$ e valor medio $\langle h\rangle$ si ottiene come $h = \langle h\rangle + z\sigma$. La generazione del numero $z$ dev'essere affidata a una funzione di nome `randomGauss()`.
(5) Le coordinate di ogni punto sono quindi salvate, immediatamente dopo essere state generate, nel *file* il cui nome è scelto dall'utente nel formato opportuno.

Esegui il programma e produci il *file*, cui assegnerai il nome `spaceGarbage.bin`. Verifica la lunghezza del *file*. Che lunghezza ti aspetti che abbia?

Una volta ottenuto il *file* passa a scrivere il secondo programma. Questo deve compiere i seguenti passi.

(1) Chiede all'utente il nome del *file* in input e prova ad aprirlo. Qualora non esista chiede nuovamente all'utente di inserirne il nome (usa il valore restituito da `fopen()` per decidere).
(2) Se il *file* esiste, il programma ne legge l'intero contenuto memorizzandolo in un array di 10 000 componenti, attraverso una sola chiamata della funzione `fread()`. Consulta il manuale di `fread()` per capire come ottenere il numero di dati effettivamente letti. Se l'operazione avrà successo, l'array conterrà $2n$ elementi, dove $n$ è il numero di detriti generati dal primo programma. Le componenti pari rappresentano la coordinata $x$ dell'oggetto e quelle dispari quella $y$.
(3) Usando la funzione `sprintf()` costruisci il nome di un *file* di output a partire da quello di input: il nome del *file* di output, che sarà un file di testo non binario, dovrà essere identico a quello di input, ma avrà estensione `gplt` (nell'esempio, `spaceGarbage.gplt`).
(4) Esegui un ciclo che, per ciascuno dei detriti generati, ne scriva sul file di output le coordinate, avendo cura di scrivere una riga per ciascun oggetto.
(5) Verifica con `gnuplot` che i punti siano distribuiti come ti aspetti. Per ottenere un'immagine il cui aspetto sia corretto usa i comandi `set xrange [-8e3:8e3]`, `set yrange [-8e3:8e3]` e `set size square`. I primi due comandi definiscono l'ampiezza della finestra (in km), mentre l'ultimo fa in modo che la finestra appaia quadrata. Per disegnare il profilo della Terra puoi usare una funzione parametrica. Per farlo puoi aggiungere il comando `set parametric`. Nel caso dell'esempio, il comando per vedere insieme il profilo della Terra e le posizioni dei detriti è `plot 6173*cos(t), 6173*sin(t), 'spaceGarbage.gplt'`. Per eliminare la legenda dalla finestra grafica puoi aggiungere, prima del comando `plot`, il comando `set nokey`.
(6) Se il controllo è positivo includi la scrittura dei dati sul *file* di output in un ciclo infinito. In questo ciclo modifica le coordinate $(x, y)$ a ogni passo, in modo tale che, al passo successivo, le coordinate siano $(x + dx, y + dy)$, dove $dx$ e $dy$ si ricavano dall'analisi dei dati memorizzati sul *file*. Una volta completata la

scrittura del *file*, il programma deve scrivere sullo schermo tutti i comandi `gnuplot` necessari per vedere il profilo della Terra e la posizione di tutti i detriti in una finestra quadrata di taglia fissata. Tra un ciclo e il successivo introduci un ritardo di un decimo di secondo.

Il calcolo di $dx$ e di $dy$ non è difficile. Basta considerare che $x = R\cos\theta$ e che $y = R\sin\theta$. Di conseguenza $dx = -R\sin\theta\, d\theta = -y\, d\theta$ e $dy = R\cos\theta\, d\theta = x\, d\theta$. Come $d\theta$ si può usare un valore costante, p.e. $d\theta \simeq 0.1$.

Eseguendo questo programma dovresti vedere sullo schermo scorrere all'infinito le seguenti righe

```
set xrange [-8e3:8e3]
set yrange [-8e3:8e3]
set nokey
set size square
set parametric
plot 6136*cos(t),6136*sin(t), 'spaceGarbage.gplt'
```

L'esecuzione del programma si può interrompere premendo contemporaneamente i tasti `Ctrl` e `C`. Il contenuto del *file* di output, naturalmente, cambia a ogni iterazione. Puoi mandare l'output di questo programma su una *pipe* che collega questo programma a `gnuplot` in modo tale che le righe scritte dal programma diventino comandi per quest'ultimo:

```
./a.out | gnuplot
```

Ricorda che all'inizio dovrai inserire il nome del *file* di input e che, se usi la *pipe*, non vedrai il messaggio corrispondente sullo schermo. Inoltre, devi considerare che, affinché `gnuplot` ignori un comando, questo dev'essere preceduto da un carattere `#`.