Laboratorio di Calcolo per Fisici, Quarta esercitazione

Canale Pb-Z, Docente: Lilia Boeri

Lo scopo della quarta esercitazione di laboratorio è di fare pratica con le istruzioni di controllo di flusso `if...(then) ...else`; `while ...do` e le funzioni di generazione di numeri casuali (*random*), scrivendo un programma che simula il gioco della roulette.

► **Prima parte:**
Scrivere un programma `lancio.c` che, utilizzando opportunamente le funzioni di generazione di numeri casuali, simuli una mano di una partita di roulette (lancio e risultato). Il programma deve:

1. Generare un numero casuale X compreso tra 1 e 36.
2. Riconoscere se il numero generato X è pari (E) o dispari (O), minore o uguale di 18 (M) o maggiore di 19 (P).
3. Stampare il risultato nel formato: X `E/O M/P`.

► **Seconda parte:** Partendo dal programma precedente, scrivere un programma chiamato `roulette.c` che invece che un singolo lancio simuli più lanci – ($N \geq 100$).

1. Alla fine degli N lanci, stampare sullo schermo la *frequenza* con cui si è verificato ciascun risultato (E/O/M/P): la frequenza è quella attesa o si discosta dal valore aspettato? Verificare come varia la frequenza al variare del parametro N - numero di lanci.
2. Costruire un *array* di 36 elementi che contenga il numero di volte in cui in un lancio si è verificato ciascun risultato tra 1 e 36.
3. Scrivere il contenuto dell'array su di un file `isto.dat` che contenga su due colonne i numeri da uno a 36 (*bin*) e le relative occorrenze. Questi dati serviranno per costruire un *istogramma* dei risultati. Il file può essere creato a mano, o redirigendo l'output del programma su un file con il comando `./programma.x > fileout`.

► **Terza parte** Far girare il programma `roulette.c` con un numero variabile di lanci, ed effettuare un'analisi dell'andamento dell'istogramma dei risultati.

1. Creare con `python` un istogramma della distribuzione dei lanci per un numero N fissato di lanci ($N \geq 100$) e salvarlo sul file su un file `isto1.gif`. Per creare un istogramma con `python` a partire da un file `dati.c` contenente due colonne (la prima contenente il numero del bin e la seconda il numero di occorreze) si usa il comando `plt.bar(x,y,fill=True)`.
2. Generare un certo numero di file di nome `istoxx.dat` che contengano l'istogramma dei risultati generati dal programma `roulette.c`, per numeri di lanci diversi: 10, 100, 10.000, 100.000. Per paragonare run con un numero diverso di lanci, invece del numero di occorrenze il file dovrà contenere la *frequenza* con cui si è verificato ciascun risultato.

3. Formattando opportunamente il grafico, paragonate gli istogrammi ottenuti per diversi valori di N. Che cosa si può dire sulla distribuzione dei risultati in funzione di N? Il risultato è corerente con le vostre aspettative? Se sì/no, perché? Salvare il grafico su un nuovo file (`isto2.gif`) e scrivere le risposte sul file `risposte.txt`.

► **Quarta parte (facoltativa)** A partire dal programma `roulette.c` creare un nuovo programma `gioco.c` che simuli una vera partita di roulette. Una partita si svolge come segue:

1. All'inizio del gioco, il giocatore riceve una dotazione iniziale di 100 euro.
2. A ogni mano, il giocatore e può decidere di fare una puntata su pari o dispari, o su un numero maggiore o minore di 18 (Manque/Passe). In caso di vincita, il giocatore riceve due volte la posta per le puntate Pari/Dispari o Manque/Passe.
3. Il gioco termina dopo un numero fissato di mani (10 o 20) o quando il giocatore esaurisce il credito a sua disposizione.

Il vostro programma dovrà simulare tutte le fasi della partita; a ogni mano dovrà chiedere al giocatore di effettuare una puntata e verificare l'eventuale vittoria, controllare se il giocatore abbia ancora soldi a propria disposizione per giocare una nuova mano. Alla fine della partita, il programma dovrà stampare un messaggio riassuntivo che riporti il numero totale delle mani giocate, l'ammontare totale delle puntate, e il credito a disposizione del giocatore.

► **Generazione di numeri casuali in C:**
Le librerie standard del C dispongono di diverse funzioni per la generazione di numeri casuali. La funzione `rand` di `stdlib.h` restituisce un numero intero casuale compreso tra 0 e il valore `RAND_MAX`, che dipende dall'architettura del processore.
► Per generare un numero casuale **compreso tra 0 e 1** bisogna dividere il risultato di `rand()` per `RAND_MAX`:
```
x=(double) rand()/RAND_MAX;
```
(L'istruzione (double) esegue il *casting* (conversione) del risultato in formato `double`.
► Per generare un numero **razionale** casuale **compreso tra 0 e N** si moltiplica l'espressione precedente per N:
```
x=(double) rand()/RAND_MAX*N;
```
(L'istruzione (double) esegue il *casting* (conversione) del risultato in formato `double`.
► Per generare un numero **intero** casuale **compreso tra 1 e N** si usa la proprietà della funzione **modulo** (`%`):
```
i=rand() % N + 1;
```
NB! Prima di chiamare la funzione `rand` all'interno di un programma bisogna inizializzare il *seme* (seed) della sequenza di numeri casuali attraverso la funzione `srand(seme)`; `seme` è un numero intero. Due sequenze inizializzate con lo stesso seme produrranno risultati identici. Se si vuole evitare che la sequenza di numeri casuali si ripeta in modo sempre uguale, si inizializza il seme utilizzando la funzione `time` della libreria nativa `time.h`. Questo si ottiene inserendo all'inizio del programma, prima della chiamata della funzione `rand`, l'istruzione:
```
srand(time(NULL));
```