Laboratorio di Calcolo per Fisici, sesta esercitazione

Canale Lp-P, Docente: Cristiano De Michele

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

▶ Prima parte:

Anzitutto creare una cartella chiamata EX6 che conterrà il materiale di questa esercitazione. Vediamo ora con alcuni esempi come si possono generare delle sequenze di numeri pseudo-casuali in C. Le librerie standard del C dispongono di diverse funzioni per la generazione di numeri casuali ma qui ci limiteremo a considerare le funzioni `rand()` e `drand48()`. Per utilizzare tali funzioni si deve includere il file `stdlib.h` all'inizio del programma con la seguente direttiva al preprocessore:

```
#include<stdlib.h>
```

La funzione `rand()` restituisce un numero intero casuale compreso tra 0 e `RAND_MAX` (che è una macro definita nel file `stdlib.h`). Quindi per generare un numero intero in un intervallo generico `[min, max]` (dove si noti che sia `min` che `max` sono inclusi) ed assegnarlo ad una variabile intera, ad es. chiamata `X`, utilizzeremo la seguente istruzione in C:

```
X = ((double)rand () / (RAND_MAX+1.0)) * (max-min+1) + min;
```

Prima di utilizzare `rand()` è necessario inizializzare il generatore di numeri random usando, una sola volta, all'inizio del vostro programma, l'apposita funzione `srand(seed)`, dove `seed` è una variabile intera positiva che contiene il "seme" del generatore. `seed` può essere un numero a caso definito nel programma ad esempio come segue:

```
#include<stdlib.h>
int main(void)
{
  int X;
  int seed=42;
  srand(seed);
  // in questo esempio sto estraendo un n. random compreso tra 0 (incluso) e 100 (incluso)
  X = ((double)rand() / (RAND_MAX+1.0)) * 101;
}
```

ed in tal caso, ad ogni esecuzione, la sequenza di numeri pseudo-casuali sarà sempre la stessa, oppure lo si può generare usando la funzione `time()` con argomento `NULL` che fornisce il tempo in secondi a partire dal 1/1/1970. In questo secondo caso il seme del generatore cambierà ogni secondo generando perciò una sequenza di numeri random sempre diversa (a meno che non si esegua il programma due volte in uno stesso secondo!). Per usare la funzione `time` bisogna includere il file d'intestazione `time.h` come nel seguente esempio:

```c
#include<stdlib.h>
#include <time.h>
int main(void)
{
  int X;
  int seed=time(NULL);
  srand(seed);
  // in questo esempio sto estraendo un numero random tra 0 (incluso) e 100 (incluso)
  X = ((double)rand() / (RAND_MAX+1.0)) * 101;
}
```

Un'alternativa all'uso della funzione `rand()` è quella di usare il generatore di numeri casuali `drand48()` che genera numeri pseudo-casuali nell'intervallo $[0, 1)$. Per generare un numero random intero nell'intervallo `[min,max]` usando `drand48()` si può utilizzare la seguente istruzione C:

```c
X = drand48()*(max-min+1)+min;
```

In questo caso il generatore di numeri random va inizializzato utilizzando la funzione `srand48()` che si usa allo stesso modo di `srand()`, ovvero fornendo un intero come argomento che costituisce il seme della sequenza di numeri pseudo-casuali. Quindi, un esempio di codice per generare numeri interi casuali tra 0 e 100 tramite `drand48()` e la funzione `time()` vista in precedenza è il seguente:

```c
#include<stdlib.h>
#include <time.h>
int main(void){
  int X;
  int min=0,max=100;
  int seed=time(NULL);
  srand48(seed);
  // in questo esempio sto estraendo un numero random compreso tra 0 e 100 inclusi
  X=(drand48()*(max-min+1))+min;
}
```

Sfruttando quanto abbiamo appena discusso ed utilizzando la funzione `drand48()`, scrivere un programma `lancio.c` che simuli una mano di una partita di roulette (lancio e risultato). Il programma deve:

1. Generare un numero casuale intero X compreso tra 1 e 36 (incluso).
2. Riconoscere se il numero generato X è pari o dispari e se è $X \leq 18$ oppure $X > 18$.
   - Se il numero è pari verrà stampata la lettera E (even)
   - Se il numero è dispari verrà stampata la lettera O (odd)
   - Se il numero è minore o uguale a 18 verrà stampata la lettera M (manque)
   - Se il numero è maggiore di 18 verrà stampata la lettera P (passe)

3. Stampare il risultato nel formato: `X E/O M/P` (cioè numero estratto pari (o dispari) minore (o maggiore) di 18). Ad esempio se uscisse il numero 19 il programma deve stampare quanto segue:

```
19 O M
```

Suggerimento: Per verificare se un numero è pari o dispari potete usare l'operatore `%`. Infatti dato un numero intero `X`, questo è pari se `X % 2 == 0`.

► **Seconda parte:** Copiare il programma `lancio.c` in `roulette.c`. Modificare `roulette.c` in modo che, invece che un singolo lancio, simuli N lanci (con $N \geq 100$). Per impostare il numero di lanci N potreste utilizzare una macro che potete definire nel codice con la direttiva del preprocessore `#define`, ovvero inserendo nel vostro codice C la seguente istruzione per il preprocessore:

```
#define N 200
```

oppure definendo la macro in fase di compilazione con il seguente comando da terminale:

```
> gcc -D N=YYY -o lancio.exe lancio.c
```

dove YYY è il numero di lanci.
Il programma dovrà fare quanto segue:

1. alla fine degli N lanci, stampare sullo schermo la *frequenza* (ossia il numero di volte in cui si è verificato un risultato diviso per numero di lanci totali) 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.
   Suggerimento: Per generare gli N numeri random dovrete utilizzare un ciclo (ad esempio un ciclo `for()`), ovvero

```
....
for(i=0; i<N; i++)
  {
    X = (drand48() * (max-min+1)) + min;
    ....
  }
....
```

   sempre all'interno del ciclo potete introdurre i controlli sul tipo di numero estratto. Usare un numero congruo di contatori inizializzati a 0 per poter contare quante volte il numero uscito è pari/dispari o minore/maggiore di 18.

2. utilizzare un array d'interi, che conterrà per ogni numero tra 1 e 36 il relativo numero di occorrenze, ovvero il numero di volte che tale numero è uscito.
   Suggerimento: L'array delle occorrenze dovrà conteggiare quante volte un certo numero random è stato estratto. Per comodità chiamiamo l'array `nocc` che deve essere necessariamente inizializzato a 0. Tale array dovrà quindi contenere questa informazione per tutti i possibili numeri compresi tra 1 e 36 e, quindi, la sua lunghezza sarà pari a 36. Ogni volta che si estrae un numero

random `X`, bisogna incrementare di 1 il contenuto della casella di memoria di `nocc` associata a quel numero `X`. In tale specifico caso potremo associare ad ogni intero estratto `X` l'elemento dell'array `X-1`, ovvero se `X` è la variabile intera che contiene l'ultimo numero pseudo-casuale generato, allora l'istruzione per aggiornare il corrispondente elemento dell'array è:

```
nocc[X-1] = nocc[X-1] + 1;
```

3. Scrivere in un file `isto.dat` su due colonne i numeri da 1 a 36 e le relative frequenze, ovvero il numero di occorrenza di ogni numero diviso per il numero totale di lanci. Questi dati serviranno per costruire un istogramma dei risultati. Il file può essere creato copiando e incollando l'output del programma in un file chiamato `isto.dat`, oppure, se `roulette.exe` è l'eseguibile del programma, ridirezionando l'output del programma su un file con il comando `./roulette.exe > isto.dat`.
   Suggerimento: dopo che tutti gli N numeri sono stati estratti, per stampare i numeri estratti e le relative frequenze potete usare un ciclo `for()` come il seguente:

```
for(i = 0; i < 36; i++)
  {
    printf("%d %f\n", i+1, (double)nocc[i]/N);
  }
```

   Se decidete di ridirezionare l'output del programma sul file chiamato `isto.dat` vi consigliamo di commentare (o di utilizzare opportunamente la compilazione condizionata) le righe di codice che stampano altre informazioni (quali ad es. quelle richieste al punto 1.) poiché in tale file devono solo essere presenti le due colonne una con i valori che vanno da 1 a 36 e l'altra con le relative frequenze.

► **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 `isto.png` (con l'istruzione python `plt.savefig('isto.png')`) Per creare un istogramma con `python` a partire da un file `isto.dat` contenente due colonne (la prima contenente il numero del bin e la seconda il numero di occorrenze) si usa il comando `plt.bar(x,y,fill=True)`, che dovrete utilizzare a posto del comando `plt.plot(...)`.

2. Generare un certo numero di file di nome `istoN.dat` che contengano l'istogramma dei risultati generati dal programma `roulette.c`, per numeri di lanci diversi, ad esempio per valori di N pari a 10, 100, 10000, 100000.

3. Formattando opportunamente i grafici, 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 coerente con le vostre aspettative? Se sì/no, perché? Scrivere le risposte nel file `risposte.txt`.

► **Quarta parte (facoltativa/per casa)** 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. Ad ogni mano, il giocatore 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. *Nota Bene:* conviene gestire gli `if` necessari per questa parte utilizzando numeri interi o singoli caratteri (per esempio E/O e M/P) invece di stringhe intere.

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 e ad ogni mano dovrà chiedere al giocatore di effettuare una puntata, verificare l'eventuale vittoria e 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. **N.B.** *Questa quarta parte è abbastanza difficile, vi consigliamo quindi di considerarla più come un esercizio da svolgere a casa in modo da potervi esercitare ulteriormente con il C.*