Corso di laurea in fisica
Anno Accademico 2020/2021
Laboratorio di informatica 1

Esercitazione 7

Nota 1 Quando un programma deve acquisire dati da un *file* strutturato in righe, ciascuna costruita secondo un formato fisso, le funzioni `fgets` e `sscanf` possono agevolare la lettura dei dati. La funzione `fgets` legge una riga da un *file*, cioè una sequenza di caratteri terminata da un carattere 'a capo', e copia i caratteri (compreso 'a capo') in un vettore. Il prototipo della funzione è questo:

```
char *fgets (char *line, int max, FILE *fp)
```

L'argomento `line` dev'essere un puntatore a un vettore di lunghezza almeno pari a `max`; la funzione legge al massimo `max-1` caratteri dalla riga corrente; i caratteri effettivamente letti vengono copiati nel vettore a cui punta `line`, e in coda viene aggiunto il segnale di fine stringa; l'argomento `fp` dev'essere un puntatore a un *file* aperto in lettura. Il carattere 'a capo' che conclude la riga viene copiato nel vettore `line`, se la riga non è stata troncata prima della fine. La funzione restituisce un puntatore al primo elemento di `line` oppure il valore `NULL` se l'operazione non è andata a buon fine.

La funzione `sscanf` si comporta come `scanf`, ma legge da una stringa di caratteri, che dev'essere passata come primo argomento.

Nota 2 Consideriamo la seguente sequenza di numeri:

```
5 2 3 4 1 2 5 6 1 2 1 6 5 2 3 2 5 2 5 6 3 4 3 2 5 2 1 4 1 2
```

Apparentemente si tratta di una sequenza "casuale", come potrebbe essere ad esempio ottenuta lanciando per trenta volte un dado. In realtà la sequenza è stata ottenuta eseguendo un semplice programma che calcola iterativamente i termini di una serie del tipo

$$x_i = (ax_{i-1} + b) \,\%\, m$$

dove $x \,\%\, y$ denota il resto della divisione di x per y. Scegliendo opportunamente i valori di a, b e m, si può fare in modo che i termini della serie appaiano casuali e, a lungo andare, distribuiti uniformemente tra 0 e $m - 1$. Queste sequenze vengono dette *pseudocasuali* e sono utili in diversi casi; ad esempio, quando si vuole simulare un fenomeno che presenta caratteri di aleatorietà, oppure nell'applicazione dei cosiddetti "metodi Monte Carlo", o quando si vuole saggiare il funzionamento di un programma su dati artificiali.

Nella libreria standard del C è compresa una funzione `rand()`, che serve per generare sequenze di numeri apparentemente casuali. A ogni chiamata, la funzione restituisce un valore intero, di volta in volta diverso, compreso fra 0 e un numero definito nella libreria stessa, e denotato da `RAND_MAX`.

NOTA: le funzioni generatrici di numeri casuali sono cruciali per molte applicazioni (ad esempio, nella simulazione con calcolatori di fenomeni fisici). In un'applicazione reale, occorrono funzioni di qualità superiore rispetto a `rand()`, ma in questo corso non è possibile approfondire l'argomento.

1 Escogitate un modo per servirvi di `rand()` allo scopo di: (1) simulare mille

lanci di due dadi, e calcolare la distribuzione statistica dei risultati; (2) generare una sequenza di cento valori di tipo `double`, compresi fra 0 e 1.

2 *Questo esercizio deve essere svolto in gruppo e consegnato entro la data della successiva esercitazione all'indirizzo* `luca.bernardinello@unimib.it`. *L'oggetto del messaggio deve consistere nella sigla LABINF, seguita dal numero di matricola di uno degli studenti del gruppo. Il file con il codice sorgente della soluzione deve avere come nome lo stesso numero di matricola indicato nell'oggetto del messaggio, seguito dall'estensione* `.c`. *La prima riga del file deve consistere in un commento che riporta i numeri di matricola dei tre componenti del gruppo, separati da spazi, come in questo esempio:*

```
// 123456 789012 345678
```

Scrivete un programma che legge K righe dallo *standard input*. Ogni riga è formata da un carattere, un numero intero e un numero reale, separati da uno spazio. Il programma deve memorizzare i dati di ogni riga in un *record*. I numeri nell'ultimo campo di ciascuna riga sono compresi fra un valore minimo a e un massimo b. Il programma deve suddividere l'intervallo $[a, b]$ in K sottointervalli di uguale ampiezza, e contare quanti valori sono compresi in ciascun intervallo. Deve infine stampare i risultati, uno per riga, in un file di nome `classi.txt`.

Il programma deve ricevere a, b e K dalla riga di comando, in quest'ordine.