Laboratorio di programmazione

20 dicembre 2007

Lunghezza media delle parole

Scrivete un programma che legga una frase (cioè, una sequenza di caratteri terminata da `.`) e che calcoli quante parole contiene e la loro lunghezza media.

Esempio di funzionamento

```
S'io fossi foco arderei il mondo.
La frase contiene 7 parole, con lunghezza media 3.71.
```

Suggerimento. Per contare quante parole contiene una stringa potete controllare quanti sono i caratteri alfabetici seguiti da caratteri non alfabetici; per vedere se un carattere è alfabetico potete usare la funzione `isalpha()` (dovete includere il file `<ctype.h>` per poterla usare).

Alfabeto farfallino

Quando i vostri docenti di laboratorio di programmazione erano bambini, usavano a volte, per comunicare con i loro simili, uno speciale alfabeto, detto *alfabeto farfallino*. L'alfabeto farfallino consiste nel sostituire, a ciascuna vocale, una sequenza di tre lettere della forma vocale-f-vocale. Per esempio, alla lettera *a* viene sostituita la sequenza *afa*, alla lettera *e* la sequenza *efe* e così via.

Dovete scrivere un programma, di nome `farf` che, ricevendo come argomento (sulla riga di comando) una parola, ne stampi la traduzione in alfabeto farfallino. Potete assumere che la stringa in input non contenga lettere maiuscole.

Parte facoltativa. Provate a modificare il programma in modo che accetti più parole sulla riga di comando.

Esempio di funzionamento

```
$./farf mamma
mafammafa
```

La strana sillabazione

Il professor Precisini, dell'*Accademia della crusca*, sostenendo che le regole di sillabazione della lingua italiana sono troppo complesse e piene di eccezioni, propone un nuovo e originale metodo di sillabazione. Il metodo consiste in questo: una sillaba è una sequenza massimale di caratteri consecutivi che rispettano l'ordine alfabetico. Per esempio, la parola *ambire* viene sillabata come *am-bir-e*: infatti la lettera *a* precede la lettera *m*, e le lettere *b*, *i* e *r* rispettano anch'esse l'ordine. Analogamente, la parola *sotterfugio* viene sillabata come *s-ott-er-fu-gio*.

Dovete scrivere un programma, di nome `sillaba` che, ricevendo come argomento (sulla riga di comando) una parola, la sillabi. Potete assumere che la stringa in input sia costituita solo da lettere minuscole.

Esempio di funzionamento

```
$./sillaba amore
amor-e
$./sillaba scafroglia
s-c-afr-o-gl-i-a
```

Il codice di Vigénère

Un sistema di cifratura molto diffuso fin dal XVI secolo è il cosiddetto *codice di Vigénère*, una variante polialfabetica del cifrario di Cesare.

Supponete di avere un *testo in chiaro*, costituito semplicemente da una sequenza di caratteri alfabetici. Per applicare il codice di Vigénère, occorre anche avere una *chiave di cifratura*, spesso chiamata "verme".

Il testo in chiaro e il verme vengono scritti uno sopra l'altro (il verme viene, se necessario, ripetuto più volte e/o troncato, in modo che le due sequenze di caratteri abbiano la stessa lunghezza). Quindi i due testi vengono sommati lettera per lettera. In pratica, questo corrisponde a identificare ogni lettera dell'alfabeto con un numero fra 0 e 25, e nell'effettuare le somme modulo 26.

Ad esempio, se il testo fosse `ARRIVANOIRINFORZI` e il verme fosse `VERME`:

```
ARRIVANOIRINFORZI
VERMEVERMEVERMEVE
VVIUZVRFUVDRWAVUM
```

Infatti A+V=V (essendo A la 0-esima lettera e V la 21-esima lettera, A+V=0+21=21=V), R+E=V (R è la 17-esima lettera e E la quarta, R+E=17+4=21=V) eccetera.

Notate inoltre che lo stesso sistema si può usare anche per decifrare un testo cifrato: è sufficiente sostituire al verme usato per cifrare quello "opposto" (sostituendo a ogni A una A, a ogni B una Z, a ogni C una Y, a ogni D una X ecc.).

Dovete scrivere un programma `vigenere` che, dopo aver ricevuto sulla linea di comando un verme (costituito solo da lettere maiuscole), legga il testo in chiaro (anch'esso costituito solo da lettere maiuscole) e stampi il testo cifrato.

Dopo aver scritto e provato il programma, provate a decodificare il seguente messaggio (sapendo che era stato codificato con il verme `CANE`, il cui opposto è `YANW`):

```
UPRVKAZSEHRGGLRWVIAEUINFWOAE
```

Esempio di funzionamento

```
$./vigenere VERME
ARRIVANOIRINFORZI
VVIUZVRFUVDRWAVUM
$./vigenere FWJOW
VVIUZVRFUVDRWAVUM
ARRIVANOIRINFORZI
```

Integrali definiti: metodo di Monte-Carlo

Considerate una funzione $f : \mathbf{R} \to \mathbf{R}$ e un intervallo $[a,b]$ sulla retta reale (con $0 \leq a < b$). Supponiamo, per semplicità, che f sia non negativa sull'intervallo $[a,b]$, cioè che $f(x) \geq 0$ per ogni x tale che $a \leq x \leq b$. L'*integrale definito di f sull'intervallo* $[a,b]$ è l'area sottesa alla curva f nell'intervallo, come appare in Figura 1.

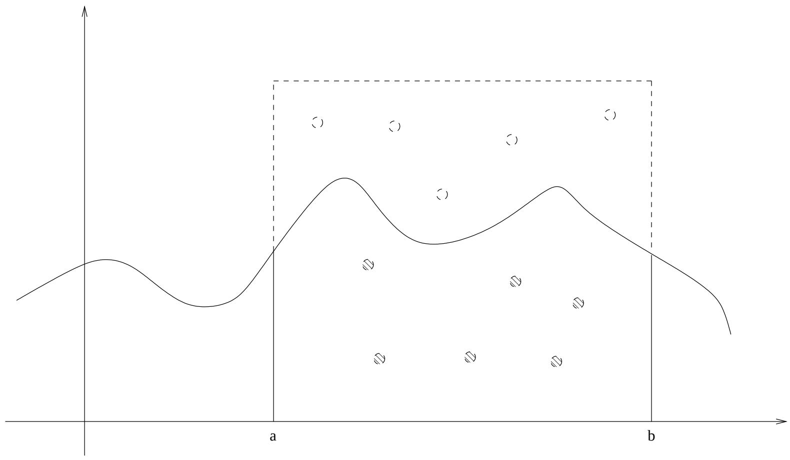

Figura 1: L'integrale definito.

Obiettivo di questo esercizio è approssimare il valore numerico dell'integrale definito utilizzando il *metodo Monte-Carlo*. Supponete di conoscere già un valore M tale che $f(x) \leq M$ per ogni x nell'intervallo $[a,b]$. Questo vuol dire che, nell'intervallo che ci interessa, il grafico della funzione è interamente confinato nel rettangolo che ha per base l'intervallo $[a,b]$ sull'asse delle ordinate, e che ha altezza M (come in figura).

Se generassimo a caso n punti in tale rettangolo, un certo numero $m \leq n$ di essi cadrà sotto la curva f, mentre i rimanenti $n-m$ cadranno sopra. È abbastanza naturale aspettarsi[1] che una stima della frazione dell'area del rettangolo che si trova sotto la curva f sia data dal rapporto m/n (e ci aspettiamo che tale stima sia tanto migliore quanto maggiore sarà il numero n di punti che useremo); una stima dell'integrale definito sarà data quindi dal prodotto tra tale rapporto e l'area del rettangolo $M(b-a)$. Nell'esempio di Figura 1 ci sono $n = 16$ punti di cui $m = 10$ sono sotto la curva (e hanno colore nero) e gli altri (di colore bianco) sono sopra la curva; se il rettangolo avesse area 2, una stima dell'integrale sarebbe $2 \times 10/16 = 1,25$.

Utilizzando questa idea, scrivete un programma per calcolare una approssimazione del valore di π (per otenere una buona approssiamzione possono essere necessari fino a un milione di punti).

Suggerimenti

Per generare punti a caso nel rettangolo usate il generatore di numeri pseudocasuali. Come osservato nella precedente lezione di laboratorio, la funzione `rand()` restituisce un numero nell'insieme $\{0,\ldots,$`RAND_MAX`$\}$, per ottenere un numero nell'intervallo $[$`l`$,$`r`$]$ potete usare la seguente trasformazione:

```
l + ( r - l ) * ( rand() / ( RAND_MAX + 1.0 ) ) )
```

Scrivete una funzione `rnd()` che dati `l` e `r` come parametri restituisca un numero a caso nell'intervallo $[$`l`$,$`r`$]$. Tramite tale funzione potete ottenere le due coordinate di un punto nel rettangolo con le chiamate `rand( a, b )` e `rnd( 0, M )`.

Assumendo che sia stata definita una funzione `f()` che dato `x` come parametro restituisca il valore $f($`x`$)$, scrivete una funzione `dint()` che dati come parametri `a`, `b`, `M` ed `n` restituisca la stima del valore dell'integrale di f sull'intervallo $[$`a`$,$`b`$]$ ottenuta con il metodo di Monte-Carlo usando `n` punti.

[1] In realtà si tratta di un risultato fondamentale della teoria della probabilità, noto come legge dei grandi numeri.

Definite ora la funzione `f()` in modo che l'integrale sia facile da calcolare (ad esempio, usate $f(x) = k$ il cui integrale sull'intervallo $[a,b]$ è $k(b-a)$ dove k è un valore a vostra scelta, ovviamente potete scegliere qualunque $M \geq k$ come altezza del rettangolo) e provate la correttezza di quanto implementato fin qui.

Osservate ora che l'area di un cerchio di raggio r è πr^2 e che $f(x) = \sqrt{r^2 - x^2}$ su $[0,r]$ rappresenta un quarto di circonferenza.