Laboratorio di programmazione

22 novembre 2007

Medie

Data una sequenza di numeri reali $x_1, x_2, \ldots, x_n$, sono dette rispettivamente

1. *media geometrica* il valore $\sqrt[n]{x_1 x_2 \ldots x_n} = \exp(\log(x_1 x_2 \ldots x_n)/n)$,
2. *media quadratica* il valore $\sqrt{\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)/n}$,
3. *media armonica* il valore $n \Big/ \left(\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}\right)$.

Scrivete un programma che legga una sequenza di numeri reali positivi in input, terminati da -1, e ne stampi le tre medie. Per scrivere il programma, potete usare le funzioni `sqrt()`, `exp()` e `log()`, che calcolano, rispettivamente, la radice quadrata, l'esponenziale e il logaritmo naturale del loro argomento. Per potere utilizzare queste funzioni, dovete inserire la dichiarazione `#include <math.h>`. Ricordatevi inoltre di compilare il programma con l'opzione `-lm`.

Suggerimento. Naturalmente, per leggere la sequenza di numeri reali dovete usare un ciclo. Durante questo stesso ciclo potete tenere traccia dei valori che vi servono per calcolare le varie medie. Per esempio, per poter calcolare la media quadratica vi servirà una variabile `n` in cui tenere il *numero* di valori letti fino a quel momento e una variabile `tot` contenente la *somma* dei quadrati dei valori letti fino a quel momento (di che tipo devono essere queste due variabili?). Ogni volta che viene letto un nuovo valore dovrete incrementare di uno la variabile `n` e incrementare la variabile `tot` del quadrato valore letto (entrambe le variabili devono essere inizializzate a zero prima del ciclo). State attenti a non incrementare né `n` né `tot` quando leggete il valore -1, che ha come unico scopo quello di terminare la sequenza degli input.

Esempio di funzionamento

```
2 4 5 -1
Media geometrica: 3.419952
Media quadratica: 3.872983
Media armonica: 3.157895
```

Espressioni ben parentesizzate

Una successione di caratteri è un'*espressione ben parentesizzata* se, per ogni prefisso della successione stessa (cioè, per ogni possibile segmento iniziale della successione), il numero di parentesi aperte "(" è maggiore o uguale al numero di parentesi chiuse ")", e se, complessivamente, il numero di parentesi aperte è uguale al numero di parentesi chiuse.

Scrivete un programma che legga (mediante la funzione `getchar()`) una sequenza di caratteri terminata da `.` e determini se essa è un'espressione ben parentesizzata. In caso negativo, il pro gramma dovrà stampare in quale posizione ha identificato un errore, e il tipo di errore.

Esempio di funzionamento

```
Stringa: ((1)abb(3(2a)4(b))5).
La stringa è un'espressione ben parentesizzata
```

```
Stringa: ((1)abb(3(2a)))4(b5.
La stringa non è un'espressione ben parentesizzata:
Carattere 19: mancano parentesi chiuse alla fine!
```

```
Stringa: ((1)abb(3))(2))a)4(b))5).
La stringa non è un'espressione ben parentesizzata:
Carattere 15: troppe parentesi chiuse!
```

Unità di $\mathbf{Z}_n$

Scrivete un programma che, dato in input un intero n, scriva le unità del gruppo $\mathbf{Z}_n$ e i loro inversi, cioè gli interi x tra 1 e $n-1$ tali che esiste un intero y che soddisfa

$$xy \equiv 1 \mod n.$$

Suggerimento. Per risolvere questo problema, dovete considerare (ovviamente, in un ciclo) tutti i valori x compresi fra 1 e $n-1$. Per ciascuno di essi (quindi, all'interno del ciclo) dovete esaminare tutti i valori di y compresi fra 1 e $n-1$ e vedere se ne esiste uno che moltiplicato per x dà 1 modulo n. In caso positivo, dovete stampare la riga corrispondente, come nell'esempio.

Esempio di input

```
9
```

Esempio di output

```
1 (inverso: 1)
2 (inverso: 5)
4 (inverso: 7)
5 (inverso: 2)
7 (inverso: 4)
8 (inverso: 8)
```

Primi gemelli

Due *primi gemelli* sono due numeri primi della forma p e $p+2$. Ad esempio, 3 e 5 sono primi gemelli. Si congettura che esistano infinite coppie di primi gemelli, ma questa congettura non è fino ad ora stata dimostrata. Scrivete un programma che, dato in input un intero n, trovi tutte le coppie $(p, p+2)$ di primi gemelli con $p \leq n$.

Esempio di output

```
n = 50
3 5
5 7
11 13
17 19
29 31
41 43
```

Suggerimento. Per determinare se un numero x è primo basta guardare tutti i valori fra 2 e $x-1$ e controllare che nessuno di essi divida x. Ovviamente, perché $(p, p+2)$ sia una coppia di primi gemelli occorre che sia p che $p+2$ siano primi.

Numeri perfetti

Un numero intero x è *perfetto* se è la somma dei suoi divisori (compreso 1 ma escluso x). Ad esempio, 28 è perfetto, essendo $28 = 1+2+4+7+14$. Euclide dimostrò che ogni numero perfetto pari è della forma $2^{n-1}(2^n-1)$; si congettura che non esistano numeri perfetti dispari (anche se questo non è mai stato dimostrato), il che porterebbe a pensare che gli unici numeri perfetti siano di questa forma. Notate, comunque, che non per tutti gli n il numero $2^{n-1}(2^n-1)$ è perfetto.

Scrivete un programma che, dato in input N, consideri tutti gli $n \leq N$ e determini se $2^{n-1}(2^n-1)$ è perfetto, stampando, in caso affermativo, sia n che il numero perfetto stesso.

Esempio di output

```
N = 15
2 6
3 28
5 496
7 8128
13 33550336
```

Numeri amichevoli

Due numeri interi (x,y) sono detti *amichevoli* se la somma dei divisori di ciascuno è uguale all'altro (fra i divisori è compreso l'1 ma è escluso il numero stesso). Ad esempio $(220,284)$ è una coppia di amichevoli, essendo

$$\begin{aligned} 284 &= 1+2+4+5+10+11+20+22+44+55+110 \text{ (che sono i divisori di 220)} \\ 220 &= 1+2+4+71+142 \text{ (che sono i divisori di 284).} \end{aligned}$$

Gli antichi greci ritenevano che le coppie di numeri amichevoli avessero certe proprietà mistiche.

Scrivete un programma che, dato in input n, stampi tutte le coppie di numeri amichevoli (x,y) con $x \leq y \leq N$.

Esempio di output

```
n = 1000
6 6
28 28
220 284
496 496
```