

Laboratorio di Algoritmi e Strutture Dati

Docente: Camillo Fiorentini

6 novembre 2007

Funzioni matematiche

Per l'esercizio 1 possono essere utili le seguenti funzioni della libreria matematica

```
double log(double x)    // restituisce il logaritmo naturale (base e) di x, dove x>0
```

```
double pow(double x, double y) // restituisce x elevato a y
```

```
double floor(double x) // restituisce la parte intera di x (max intero n tale che n<=x)
```

Calcolo dei logaritmi

Con la funzione `log()` è possibile calcolare il logaritmo in una qualunque base. Infatti, se $b > 0$, $b \neq 1$ e $a > 0$ vale:

$$\log_b a = \frac{\ln a}{\ln b}$$

dove $\ln$ è il logaritmo naturale.

Nota

Ricordarsi che con alcune versioni di `gcc` occorre usare l'opzione `-lm`, altrimenti il linker non trova il codice oggetto delle funzioni della libreria matematica.

Esercizio 1

Per risolvere l'esercizio, completare il file `printInt.c`.

a. Scrivere una funzione

```
void print10(int n)
```

che stampa la rappresentazione in base 10 di un intero $n \geq 0$ usando solo `putchar()`.

Ragionare per induzione sul numero di cifre di n in base 10.

- *Caso base.*

Se n si rappresenta con una cifra, si può immediatamente stampare n usando `putchar()`.

- *Passo induttivo.*

Supponiamo che n si rappresenti con $k \geq 2$ cifre.

Per *ipotesi di induzione*, si può assumere che `print10(m)` stampi correttamente gli interi $m \geq 0$ che hanno *meno* di k cifre.

Si possono allora determinare due interi n_1 e u tali che la rappresentazione in base 10 di n_1 corrisponde alle *prime* $k - 1$ cifre di n , e u è l'ultima cifra di n .

Ad esempio, se $n = 1234$, allora $n_1 = 123$ e $u = 4$.

Per l'ipotesi di induzione fatta prima, si possono stampare sia n_1 che u mediante una chiamata ricorsiva a `print10()` (infatti, n_1 ha $k - 1$ cifre, u ha una cifra). Stampando n_1 e u (nel giusto ordine), si ottiene la stampa di n .

b. Se non si usa la ricorsione, la soluzione è più complessa. Infatti, mentre è immediato determinare l'ultima cifra della rappresentazione in base 10 di un intero n , per determinare la prima cifra da stampare occorre prima sapere quante cifre ha n .

Scrivere una funzione

```
int length10(int n)
```

che restituisce il numero di cifre della rappresentazione in base 10 di n ($n \geq 0$).

Ad esempio, se n vale 325, la funzione deve restituire 3.

Scrivere tre diverse versioni di `length10()`:

(b1) *Versione ricorsiva.*

Il caso base si ha quando n ha una cifra.

Se n ha $k \geq 2$ cifre, si può determinare (come nel punto a) l'intero n_1 ottenuto da n eliminando l'ultima cifra e calcolare ricorsivamente il numero di cifre di n_1 (esempio, se $n = 734$, allora $n_1 = 73$).

(b2) *Versione iterativa.*

Scrivere un ciclo in cui ad ogni iterazione si elimina l'ultima cifra di n , finché si ottiene un numero di una cifra. Dal numero di iterazioni effettuate si può determinare il numero di cifre di n .

Ad esempio, supponiamo $n = 7365$.

- Alla prima iterazione, elimino 5 (ultima cifra di n) e n vale 736.
- Alla seconda iterazione elimino 6 (ultima cifra di n) e n vale 73.
- Alla terza iterazione elimino 3 (ultima cifra di n) e n vale 7.

Avendo ottenuto un numero di una cifra, il ciclo termina e si può affermare che il numero n inizialmente dato ha 4 cifre.

(b3) *Calcolo diretto.*

Calcolare il numero di cifre di n usando una formula matematica.

c. Supponiamo di numerare le cifre della rappresentazione in base 10 di n dalla meno significativa alla più significativa (in C è consuetudine iniziare a contare da 0).

Ad esempio, se $n = 735$:

- La cifra in posizione 0 è 5.
- La cifra in posizione 1 è 3.
- La cifra in posizione 2 è 7.

Quindi, l'ultima cifra di n (la cifra meno significativa) ha posizione 0, la prima cifra di n (cifra più significativa) ha posizione $length10(n) - 1$.

Scrivere una funzione

```
int pos10(int n, int k)
```

che restituisce l'intero corrispondente alla cifra in posizione k della rappresentazione in base 10 di n , dove si assume che $0 \leq k < \text{length10}(n)$.

d. Usando le funzioni `length10()` e `pos10()`, scrivere una funzione iterativa

```
void print10it(int n)
```

che stampa la rappresentazione in base 10 di un intero $n \geq 0$ usando solo `putchar()`.

e. Riformulare i punti precedenti considerando le rappresentazioni in base 2 e in base 16 di n . Occorre quindi definire il codice delle funzioni

```
void print2(int n)      int length2(int n)      int pos2(int n,int k)      void print2it(int n)
```

```
void print16(int n)     int length16(int n)     int pos16(int n,int k)     void print16it(int n)
```

il cui significato è analogo a quello di prima.

In realtà tutte le funzioni viste sopra possono essere scritte in forma generica, passando la base b come ulteriore parametro delle funzioni e assumendo $2 \leq b \leq 16$. Se si vogliono usare basi più grandi di 16, occorre definire come vanno rappresentati gli interi n tali che $16 \leq n < b$, ossia occorre aggiungere nuove cifre a quelle già usate nella rappresentazione in base 16.

Esempi di valori

Base 10	Base 2	Base 16
8	1000	8
11	1011	b
125	1111101	7d
171	10101011	ab
678	1010100110	2a6
1024	10000000000	400

Esercizio 2

L'obiettivo è di completare il programma `calc.c` che esegue operazioni elementari su frazioni. Il file `calc.c` contiene le funzioni viste qualche lezione fa e il codice di `main()`.

Il *formato* per l'input e l'output è quello utilizzato nei **progetti d'esame**. Si *assume* che l'input sia conforme alle specifiche descritte nella tabella sottostante, per cui non è richiesto di controllare che i dati siano inseriti correttamente. Per provare il programma, conviene preparare un file di input (vedi file `in.txt`).

Il programma deve leggere da standard input una sequenza di linee (separate da '`\n`'), ciascuna delle quali corrisponde a una linea della prima colonna della tabella, dove a , b , c , d sono interi, con $b \neq 0$ e $d \neq 0$. I vari elementi sulla linea sono separati da uno o più spazi. Quando una linea è letta, viene eseguita l'operazione associata e stampato il risultato; le operazioni di stampa sono effettuate sullo standard output e ogni operazione deve iniziare su una nuova linea.

Istruzione in input	Operazione
<code>s a b</code>	Semplifica la frazione $\frac{a}{b}$ e stampa il risultato ottenuto
<code>+ a b c d</code>	Calcola $\frac{a}{b} + \frac{c}{d}$ e stampa il risultato in forma semplificata
<code>- a b c d</code>	Calcola $\frac{a}{b} - \frac{c}{d}$ e stampa il risultato in forma semplificata
<code>* a b c d</code>	Calcola $\frac{a}{b} \times \frac{c}{d}$ e stampa il risultato in forma semplificata
<code>: a b c d</code>	Se $c \neq 0$, calcola $\frac{a}{b} : \frac{c}{d}$ e stampa il risultato in forma semplificata Se $c = 0$, stampa un messaggio di errore.
<code>? a b c d</code>	Verifica se $\frac{a}{b} < \frac{c}{d}$ oppure $\frac{a}{b} = \frac{c}{d}$ oppure $\frac{a}{b} > \frac{c}{d}$ e stampa il risultato del confronto
<code>f</code>	Termina l'esecuzione del programma

Nota

Fra le funzioni già scritte in `calc.c` occorre solamente usare

```
void simplify(int x, int y, int* p, int* q)
```

tale che, dati due interi x e y , con $y \neq 0$, $\frac{*p}{*q}$ è la forma semplificata di $\frac{x}{y}$.