Laboratorio di algoritmi e strutture dati
Primi programmi

Docente: Violetta Lonati

Giovedì 28 settembre 2017

1 Somma di numeri

Scrivete un programma che legga una serie di numeri positivi terminata da 0 e ne calcoli la somma. Per leggere un intero e memorizzarlo nella variabile n potete usare l'istruzione `scanf( "%d", &n );`

Notate che non c'è motivo per tenere in memoria tutti i numeri letti da standard input, è sufficiente tenere in memoria l'ultimo letto!

2 Media di numeri

Modificate il programma precedente in modo da calcolare la media dei numeri inseriti.

3 Somma di *esattamente* 10 numeri non nulli

Scrivete un programma che legga una serie di numeri e ne calcoli la somma; il programma deve terminare dopo aver letto esattamente 10 numeri diversi da 0. Può essere utile l'utilizzo delle istruzioni `continue` e/o `break`.

4 Indovina il numero

Il gioco *Indovina il numero* funziona come segue: un giocatore A pensa a un numero intero x (con $0 \leq x \leq 1000$), e l'altro giocatore, B, lo deve indovinare. Per farlo, B pone domande del tipo "Il numero è y?", cui A può rispondere $=$ (per indicare che il numero è stato indovinato), oppure $<$ (per indicare che x è minore del numero y), oppure $>$ (per indicare che x è maggiore del numero y).

Scrivete un programma che giochi come giocatore B, seguendo questa strategia: gli estremi entro cui può stare il valore da indovinare vengono memorizzati in due variabili che verranno aggiornate ad ogni risposta dell'utente; ogni volta, il programma chiede se il valore è quello in mezzo all'intervallo e, a seconda della risposta dell'utente, il programma esce oppure modifica i valori di un estremo in modo da restringere l'intervallo.

Per la lettura delle risposte dell'utente giocatore A si consiglia di usare la funzione `scanf` con questa specifica di formato: `" %c"` (*con uno spazio prima di* `%c`).

Esempio di funzionamento:

```
Il numero è 500? <
Il numero è 249? <
Il numero è 124? <
Il numero è 61? <
Il numero è 30? >
Il numero è 45? <
Il numero è 37? =
```

NOTA: Questo algoritmo di ricerca vale per ogni sequenza ordinata ed è chiamato algoritmo di *ricerca dicotomica*.

5 Il cifrario di Cesare

Svetonio nella *Vita dei dodici Cesari* racconta che Giulio Cesare usava per le sue corrispondenze riservate un codice di sostituzione molto semplice, nel quale la lettera chiara veniva sostituita dalla lettera che la segue di tre posti nell'alfabeto: la lettera A è sostituita dalla D, la B dalla E e così via fino alle ultime lettere che sono cifrate con le prime.

Più in generale si dice codice di Cesare un codice nel quale la lettera del messaggio chiaro viene spostata di un numero fisso k di posti, non necessariamente tre.

Dovete scrivere un programma che legga da input il numero k (chiave di cifratura) e il testo da cifrare, sotto forma di una sequenza di caratteri terminata da `.` e che emetta in output il testo cifrato; il programma deve cifrare solo le lettere dell'alfabeto, mantenendo minuscole le minuscole, e maiuscole le maiuscole, mentre deve lasciare inalterati gli altri simboli.

Si consiglia di usare `scanf(%d)` per leggere k e la funzione `getchar()` per leggere uno a uno i caratteri del testo da cifrare.

Notate che non c'è motivo per tenere in memoria tutti i caratteri del testo da cifrare, è sufficiente tenere in memoria l'ultimo letto!

Per provare se il programma funziona, cifrate un messaggio con una certa chiave k e poi applicate al risultato una nuova cifratura con chiave $26 - k$: il risultato dovrebbe essere la stringa originale.

Esempio di funzionamento

```
5 mamma li Turchi
rfrrf qn Yzwhmn
```

```
21 rfrrf qn Yzwhmn
mamma li Turchi
```

6 Calendario

Scrivete un programma che stampi un calendario mensile. L'utente deve specificare il nome del mese e il giorno della settimana in cui comincia il mese. Per semplicità considerate solo anni non bisestili...

Esempio di funzionamento:

```
Inserisci il numero del mese (1 = gennaio, ···, 12 = dicembre): 2
Inserisci il giorno di inizio mese (1 = lunedi, ···, 7 = domenica): 4
 L   M   M   G   V   S   D
             1   2   3   4
 5   6   7   8   9  10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28
```

7 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. Questo è ciò che avviene, per esempio, nelle espressioni aritmetiche ben formate.

Scrivete un programma che legga (mediante la funzione `getchar()` inclusa in `stdio.h`) una sequenza di caratteri terminata da `.` e determini se essa è un'espressione ben parentesizzata. In caso negativo, il programma 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!
```

8 Disegni

Questo esercizio richiede di scrivere dei programmi che stampino delle "figure", secondo gli schemi sotto definiti. L'utente dovrà inserire un intero n che definisce la dimensione della figura da disegnare.

1. Righe alternate - la figura è ottenuta alternando righe costituite solo da + e righe costituite solo da o:

```
n = 3
+ + +
o o o
+ + +
```

```
n = 5
+ + + + +
o o o o o
+ + + + +
o o o o o
+ + + + +
```

2. Caratteri alternati: la figura è ottenuta alternando caratteri o e +:

$n = 5$

```
o + o + o
+ o + o +
o + o + o
+ o + o +
o + o + o
```

$n = 6$

```
o + o + o +
+ o + o + o
o + o + o +
+ o + o + o
o + o + o +
+ o + o + o
```

3. Triangolo: il triangolo sotto la diagonale con direzione alto/sx verso basso/dx è formato da o, il triangolo sopra la diagonale da +, la diagonale da |.

$n = 5$

```
| + + + +
o | + + +
o o | + +
o o o | +
o o o o |
```

$n = 6$

```
| + + + + +
o | + + + +
o o | + + +
o o o | + +
o o o o | +
o o o o o |
```

4. Spunta - la figura riproduce il simbolo di spunta su uno sfondo di puntini (il ramo di sinistra è formato da 3 asterischi e parte dal bordo sinistro dello schermo, il ramo di destra è formato da *n* asterischi):

$n = 5$

```
......*
.....*.
*...*..
.*.*...
..*....
```

$n = 8$

```
.........*
........*.
.......*..
......*...
.....*....
*...*.....
.*.*......
..*.......
```