Laurea In Ingegneria dell'Informazione

Esercitazioni Guidate di Tecniche della Programmazione

Note introduttive: Seguire le raccomandazioni date nelle precedenti EG …

9. Esercitazione 9 - ricorsione

9.1. La funzione fattoriale(FATT.C)

Scrivere una funzione `int fatt(int)` che, ricevendo un numero intero `n` calcola e restituisce il valore di `n!`.

Scrivere inoltre un programma capace di testare questa funzione e verificare che, usando nella definizione della funzione il tipo int (per parametro e valore ritornato) la funzione produce risultati troppo grandi anche con numeri relativamente piccoli. (Fino a che valore di `n` si riesce ad arrivare senza andare in *overflow*?)

9.2. La funzione fattoriale (FATTLONG.C)

Ripetere l'esercizio precedente, ma usando il tipo `long int` per il risultato della funzione. Verificare che questa funzione permette di fare calcoli su numeri (un pò) più grandi rispetto alla precedente.

9.3. La funzione LeggiEInvertiInput() (INVINP.C)

Scrivere una funzione che esegue la lettura di una sequenza di caratteri fornita da input e terminata dal carattere '.', producendo in output una stampa dei medesimi caratteri (escluso il punto) in ordine inverso rispetto a quello con ui sono stati inseriti in input. Ad esempio, se l'input fosse TOPI. l'output dovrebbe essere IPOT

Si vuole che la funzione implementi un algoritmo ricorsivo per risolvere il problema. Provare la funzione con un opportuno programma.

…

Suggerimento:

L’idea di quanto visto a lezione era la seguente: l’algoritmo da realizzare deve

- leggere il primo carattere che si presenta in input e memorizzarlo in una variabile `ch`;

  (ad esempio T viene messo in `ch`)
- chiedere che il resto dell’input venga letto e stampato al contrario
- se l’operazione indicata al punto precedente e’ stata realizzata, ora in output ci sono, stampati in ordine inverso, i caratteri che erano forniti in input dopo quello che abbiamo memorizzato in `ch`

  ad esempio ora in output c’è **IPO**

  quindi è sufficiente stampare il carattere attualmente in ch e l’output assume la forma richiesta

  IPO**T**

(seguono altri suggerimenti)

Suggerimento 2 di 3:

Chiamiamo LEGGIEINVERTI la sequenza di operazioni che vogliamo implementare (insomma, l'algoritmo ...)

Lo schema base da realizzare e' il seguente:

- lettura di un carattere in `ch`
- LEGGIEINVERTI
  (così viene letto e invertito il resto dell'input (quello dopo il carattere memorizzato in ch)
- stampa `ch`

Suggerimento segue

Suggerimento 3:

Ovviamente se il carattere letto e memorizzato in `ch` durante l'esecuzione di LEGGIEINVERTI e' `'.'`, non deve essere richiesto di leggere e invertire "il resto dell'input" (non c'e'!), ne' di stampare il carattere `ch`.
Queste due operazioni vanno eseguite solo se il carattere letto è diverso da `'.'`.

Quindi il caso in cui `ch!='.'` è un CASO RICORSIVO, in cui l'algoritmo LEGGIEINVERTI decide di eseguire se stesso sul resto dell'input e poi stampare ch.

Invece il caso in cui `ch=='.'` è un CASO BASE (in cui non c'è attivazione ricorsiva e, in questo caso particolare, non bisogna fare nulla).

```
- lettura ch
- se ch!='.'
     o LEGGIEINVERTI
     o stampa ch
  (altrimenti niente)
```

Poi segue il prossimo esercizio

9.4. La funzione palin() (PALIN1.C)

Scrivere una funzione palin() che, ricevendo una parola (stringa di caratteri) e due indici interi, i, j, restituisca 1 se la porzione di parola compresa tra i caratteri di indice i e j è palindroma.

Provare la funzione con un opportuno programma.

Attenzione, segue suggerimento …

Suggerimento:

Una parola è palindroma se è leggibile indifferentemente da sinistra a destra o da destra a sinistra (anilina, asor-rosa).

Attenzione, segue suggerimento …

Suggerimento 2 di 5:

Sia p la parola (un array di N caratteri, occupato dal carattere di indice 0 al carattere di indice `k` (con `strlen(p)` uguale a `k+1` e `p[strlen(p)]=='\0'` e quindi escluso dalle nostre considerazioni).

La porzione `p[0]---p[k]` (cioe' la stringa)

- non è palindroma se `p[0]!=p[k]`
- **potrebbe** essere palindroma se `p[0]==p[k]`(e in tal caso sarebbe effettivamente palindroma **solo se** fosse palindroma la porzione `p[1]---p[k-1]`)

Attenzione, segue suggerimento …

Suggerimento 3 di 5:

Dal suggerimento precedente viene uno schema come il seguente

`palin(p, i, j)` **è** (ritorna come risultato)

- 0 se `p[0]!=p[k]`
- il valore di `palin(p, i+1, j-1)` se `p[0]==p[k]`

Segue suggerimento

Suggerimento 4 di 5:

Lo schema visto al punto precedente vale se i<j;
infatti,

- se `i==j` non c'è nulla da controllare! E se siamo arrivati ad eseguire la valutazione di `palin(p,i,j)` con `i==j` vuol dire che tutte le coppie `p[i], p[j]` con i valori precedenti di `i` e `j` erano a caratteri uguali. A questo punto la stringa si rivela palindroma e possiamo restituire 1 senza ulteriori attivazioni ricorsive;

- se `i>j`, vale un discorso analogo: tutte le coppie `p[i], p[j]` con i valori precedenti di i e j (cioè con valori per cui `i<j`) erano a caratteri uguali. Continuando a fare verifiche su coppie di caratteri `p[i], p[j]` con `i > j` controlleremmo coppie già controllate (e che erano uguali). Quindi anche in questo caso la stringa si e' rivelata palindroma e possiamo restituire 1 senza ulteriori attivazioni ricorsive.

Segue suggerimento

Suggerimento 5:

Dal suggerimento precedente viene uno schema come il seguente

`palin(p, i, j)` **e'** (ritorna come risultato)

- se `i<j`
  - **`0`** se `p[0]!=p[k]`
  - **`palin(p, i+1, j-1)`** se `p[0]==p[k]`
- altrimenti **`1`**

9.5. cosa fa?

Osservare il seguente programma:

- la main() contiene due errori, facilmente evidenziabili. Quali sono?
- cosa fa la funzione `cosaFa()` e, quindi, cosa fa il programma, una volta corretto? Giustificare la risposta **e poi** vedere se è giusta, eseguendo il programma.

```
#include <stdio.h>
#define MAX 6

int cosaFa(int a[], int k, int dim) {
     if (k < dim)
          return (a[k] + cosaFa(a, k+1, dim));
     else return 0;
}

int main(){
     int io[MAX], j;
     printf("\nInserire le %d cifre del numero di matricola
                                        (uno spazio tra l'una e l'altra): ", MAX);

     for(j=0; j<=MAX; j++)
          scanf("%d",io+j);

     printf("ecco qua: %d\n", cosaFa(io,2,MAX));
     printf("\nFINE\n");
return 0;
}
```