Esercizi di programmazione per il corso di Algoritmi*

1 Lezione 2

Esercizi sull'esempio

Si realizzi il codice dell'esercizio "catalogo" applicando la progettazione *top-down* anziché *bottom-up*, vale a dire:

1. affrontando direttamente il problema complessivo (lettura di un catalogo da file di testo il cui nome venga indicato nella linea di comando);
2. dividendolo in sottoproblemi, e questi a loro volta in sottoproblemi ancora più semplici;
3. risolvendo ciascuna fase con chiamate a funzioni che risolvano i problemi di livello inferiore, in modo che il codice sia sempre sintatticamente corretto.

Esercizi sul *parsing*

Esercizio 1 Si supponga di eseguire l'istruzione `n = scanf("%d%f%d",&i,&r,&j);` dove `i`, `j` e `n` sono variabili intere e `r` è una variabile reale. Supponendo di aver scritto da tastiera i seguenti ingressi, si indichi il valore delle quattro variabili e il contenuto dello *stream* di ingresso dopo la chiamata.

- `10 20 30`
- `1.0 2.0 3.0`
- `0.10 0.20 0.30`
- `a.1 .2 3`

Esercizio 2 Si scriva un programma MAGAZZINO.C che riceve da tastiera un articolo alla volta, nel formato

articolo prezzo giorno anno

dove *articolo* è una singola parola di lunghezza massima pari a 10, *prezzo* è un numero reale minore di 1000.0, *giorno* e *anno* sono numeri interi che indicano il giorno e l'anno di una data e *mese* è il nome di uno dei dodici mesi (in minuscolo). Il termine del catalogo di articoli viene rappresentato dall'inserimento da tastiera di una riga contenente solo la parola chiave `STOP`. Si può dare per scontato che non vi siano più di 10 articoli.

*tratti o ispirati dal testo di K.N. King

Il programma deve poi stampare a video il catalogo in tre colonne, rispettivamente di 10, 9 e 11 caratteri, secondo il formato dell'esempio seguente:

```
Articolo  Prezzo    Data
Articolo1     12.50 21/05/2010
Articolo2    199.90 01/04/2010
```

Esercizio 3 Si scriva un programma DURATA.C che riceve da tastiera due orari, nel formato

ora1 : *minuto1* *ora1* : *minuto2*

con un numero qualsiasi di spazi fra loro, e restituisce la differenza in minuti fra i due orari.

Esercizio 4 Si scriva un programma ESTENSIONE.C che riceve da tastiera il nome di un file, ne cambia l'estensione in `.txt` e restituisce a video il risultato. Si definisca estensione quella parte del nome del file che sta dopo l'ultimo punto. Si tenga conto del fatto che il nome originale del file può contenere più di un punto, ma anche nessun punto.

Suggerimento: si può costruire il risultato sia con la funzione `strcat` sia con la funzione `sprintf`.

Esercizio 5 Si scriva un programma PARAM.C che riceve da linea di comando una qualsiasi sequenza di parametri e la stampa a video, una riga alla volta, nel formato:

```
printf("Il parametro n.  %d e' uguale a %s\n",p,argv[p]);
```

Si sostituisca `%d` con:

- `%2d` per avere una larghezza minima di 2 cifre: per numeri da 1 cifra, questo significa aggiungere uno spazio PRIMA del numero (con 10 o più parametri, questo consente di avere un allineamento perfetto dell'uscita)
- `%-2d` per allineare i numeri a sinistra anziché a destra
- `%02d` per aggiungere uno 0 prima del numero anziché uno spazio

Esercizio 6 Si scriva un programma DATA.C che riceva da linea di comando una data nel formato

`GG/MM/AAAA`

dove `GG` è un numero di due cifre che indica il giorno, `MM` un numero di due cifre che indica il mese e `AAAA` un numero di quattro cifre che indica l'anno, e la trasforma nella corrispondente data in formato letterale, cioè con il mese scritto a parole, verificando che i tre parametri siano effettivamente tre numeri, che siano compresi negli intervalli corretti (da 1 a 12 i mesi, da 1 a 28, 29, 30 o 31 i giorni, ecc. . .).

Esercizi sui parametri del `main`

Esercizio 1 Scrivere un programma `LUNGHEZZE.C` che stampa a video i parametri della linea di comando, escluso il nome del programma stesso, con la loro lunghezza.

Variante più complessa: scrivere un programma che stampa a video i parametri in ordine di lunghezza crescente, sfruttando gli algoritmi di ordinamento descritti nella lezione sulla ricorsione.

Esercizio 2 Scrivere un programma `SOMMA.C` che stampa a video la somma dei numeri interi passati come parametri nella linea di comando.

Esercizio 3 Scrivere un programma `ESEGUI.C` che riceve da linea di comando una somma o un prodotto nel formato `n1 + n2` o `n1 x n2`, con `n1` e `n2` valori interi, riconosce l'operazione da eseguire e ne restituisce il risultato corretto[1].

Esercizi sulla gestione dei file

Esercizio 1 Si scriva il codice LUNGHEZZA_RIGA.C, che riceve da linea di comando il nome di un file di testo e un numero intero `n` e stampa a video la lunghezza (numero di caratteri escluso l'a capo finale) della riga `n`-esima di tale file. Se il file contiene meno di `n` righe, si deve stampare 0.

Esercizio 2 Si scriva il codice APRIBILE.C, che riceve da linea di comando un numero qualsiasi di nomi di file di testo e stampa a video l'elenco di quelli che si possono aprire e di quelli che non si possono aprire. Il codice deve restituire al sistema operativo il valore `EXIT_SUCCESS` se tutti i file si possono aprire, `EXIT_FAILURE` se almeno uno dei file non si può aprire.

Esercizio 3 Si scriva il codice MAIUSCOLO.C, che riceve da linea di comando il nome di un file di testo e stampa a video il file stesso, convertendo tutte le lettere minuscole in maiuscole. Si considerino le due varianti nelle quali si legge una parola alla volta o una riga alla volta.

Esercizio 4 Si scriva il codice CONCATENA.C, che riceve da linea di comando i nomi di diversi file di testo esistenti e di un file da creare (l'ultimo) e salva nell'ultimo file la concatenazione del contenuto dei file iniziali, nell'ordine con cui compaiono nella linea di comando. Si usi un solo puntatore a file e l'apertura in accodamento.

Esercizio 5 Si scriva il codice LEGGEPROBLEMA.C, che riceve da linea di comando il nome di un file di testo, che contiene i dati per il seguente problema: una fabbrica produce n prodotti utilizzando m componenti. Ciascuna unità del prodotto j garantisce un profitto p_j, ma consuma una quantità a_{ij} del componente i. In magazzino sono presenti b_i unità del componente i. I dati n, m, p_j, a_{ij} e b_i

[1] La moltiplicazione va indicata con il carattere `x` anziché il consueto `*` perché in ambiente Linux (in teoria, non in ambiente Windows) il carattere `*` viene usato dal sistema operativo per indicare la lista dei file contenuti nella cartella, e viene convertito in tale lista prima di passarlo all'eseguibile, che quindi non riceve due interi separati da un operatore, ma due interi separati da una lista di nomi di file.

sono rappresentati secondo le specifiche del linguaggio di modellazione *MathProg*. Un esempio di tale linguaggio è riportato qui di seguito:

```
param NumProdotti := 3 ;
param NumComponenti := 4 ;

param Profitto :=
1 50
2 30
3 30
;

param Consumo :=
[1,1] 2 [1,2] 1 [1,3] 1
[2,1] 1 [2,2] 0 [2,3] 1
[3,1] 0 [3,2] 1 [3,3] 1
[4,1] 1 [4,2] 2 [4,3] 1
;

param Magazzino :=
1 1000
2 400
3 700
4 1200
;

end;
```

Il codice da scrivere deve caricare i valori di n e m in opportune variabili, quelli di p_j e b_i in opportuni vettori e quelli di a_{ij} in un'opportuna matrice. Vettori e matrice devono essere allocati dinamicamente. Quindi il programma deve stampare i dati a video.

Suggerimento: Si veda anche la dispensa `Leggere.pdf` sul sito del corso di programmazione.