Esercizi per il settimo laboratorio

Antonio Miele Ettore Speziale Michele Tartara

23 novembre 2010
25 novembre 2010
26 novembre 2010

Percorso Manhattan - `mpath`

Scrivere un programma che calcola la lunghezza di un percorso definito da una serie di punti letti dal file allegato (`points.dat`).

Come distanza tra due punti p e q si consideri la distanza di Manhattan:

$$d(p, q) = |p_x - q_x| + |p_y - q_y|$$

Una volta calcolata la lunghezza dell'intero percorso, il programma la stampa a video.

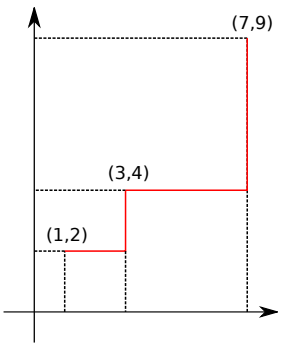
Note

I punti sono definiti come coppie di numeri interi, e sono rappresentati nel file con il formato `(%d,%d)`.

Sia Figura 1 un esempio di percorso.

```
(1,2)(3,4)(7,9)
```

(a) contenuto del file



(b) rappresentazione grafica

Figura 1: esempio di percorso.

Esso definisce un percorso composto da due passi: il primo collega $(1, 2)$ a $(3, 4)$, mentre il secondo permette di raggiungere $(7, 9)$ partendo da $(3, 4)$.

Siano p, q e r i tre punti; allora:

$$d(p, q) = |1 - 3| + |2 - 4| = 4$$
$$d(q, r) = |3 - 7| + |4 - 9| = 9$$
$$len(path) = d(p, q) + d(q, r) = 13$$

Cioè, la lunghezza del percorso completo è 13 unità.

Stack di parole - `wstack`

Il file allegato, `words.dat`, contiene una lista di parole. Scrivere un programma che:

1. richiede all'utente di inserire 3 caratteri;
2. legge dal file le parole che iniziano con uno qualsiasi dei caratteri inseriti, le raggruppa per iniziale e le salva in un array;
3. stampa le parole lette in ordine inverso rispetto all'iniziale e all'ordine di lettura.

Si considerino i seguenti limiti:

lunghezza parole ogni parola presente nel file è composta da al massimo 20 caratteri;

numero di parole il programma considera al massimo 10 parole per ogni iniziale; ulteriori parole vengono scartate;

numero di iniziali il programma considera 3 iniziali.

Esempio

Siano `c`, `p`, `m` i tre caratteri inseriti dall'utente (in questo ordine). Sia Figura 2 il contenuto del file.

```
mucca pollo ciao pippo erba nonno micio cane lumanca
```

Figura 2: contenuto del file `words.dat`.

Il programma considera solo le parole che iniziano per:

- `c`: `ciao`, `cane`;
- `p`: `pollo`, `pippo`;
- `m`: `mucca`, `micio`.

Esse vengo stampate in ordire inverso rispetto all'iniziale e all'ordine di lettura:

- stampa delle parole che iniziano con `m` in ordine inverso;
- stampa delle parole che iniziano con `p` in ordine inverso;
- stampa delle parole che iniziano con `c` in ordine inverso.

L'output del programma sarà quindi:

```
micio mucca pippo pollo cane ciao
```

Note

Per rappresentare gli array di parole raggruppate per iniziali avete due alternative.

Array tridimensionale

Usate un array a tre dimensioni:

- iniziale;
- parola;
- carattere.

Quindi, dato l'array:

```
char matches[3][10][20];
```

Posso accedere a:

- `matches[i]`: parole con la i-esima iniziale;
- `matches[i][j]`: j-esima parola con la i-esima iniziale;
- `matches[i][j][k]`: k-esimo carattere della j-esima parola con la i-esima iniziale.

Array di strutture

Usate un array di `match_t`:

```
match_t matches[3];
```

Il suo i-esimo elemento corrisponde alle parole con la i-esima iniziale.
Il tipo `match_t` è una `struct`:

```
typedef struct {
  char first;
  char words[10][20];
} match_t;
```

Il campo `first` rappresenta l'iniziale, mentre il campo `words` contiene tutte le parole che iniziano con `first`.

Avvertenze

Lo scopo dell'esercizio è di farvi utilizzare strutture dati relativamente complesse, quindi `non utilizzate un array diverso per ogni iniziale`.