APPELLO SCRITTO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2014–2015

15.VI.2015

VINCENZO MARRA

INDICE

Avvertenza. Il tema d'esame richiede lo sviluppo di un singolo programma che computa alcuni dati relativi a un insieme di triangoli nel piano memorizzati in un file. I triangoli sono acquisiti dal file `dati.txt` allegato al tema d'esame. Svilupperete alcune funzioni ausiliarie nel primo esercizio. Integrerete poi le funzioni ausiliarie in un unico programma finale, che consisterà di un unico file. (Il tema d'esame richiede esplicitamente l'implementazione di alcune funzioni ausiliarie. È comunque sempre ammissibile aggiungere altre funzioni ausiliarie qualora lo riteniate opportuno per una migliore implementazione della vostra soluzione.) Si raccomanda di leggere interamente il tema d'esame prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale.

ESERCIZIO 1

Definizione dei tipi `P` e `T`, e calcolo dell'area.

Ultima revisione: 16 giugno 2015.

Punti: 8.

Usando `typedef` e `struct`, definite i nuovi tipi `P` e `T` i cui valori rappresentano, rispettivamente, punti e triangoli del piano cartesiano. Per i valori delle coordinate usate il tipo `double`. I triangoli sono rappresentati dai loro tre vertici.

Dichiarate e implementate poi una funzione di nome `area` che accetti in ingresso un puntatore a `T` e restituisca in uscita il valore dell'area del triangolo in argomento. Per il calcolo dell'area, implementate una funzione ausiliaria

`det`

di prototipo appropriato, che accetti in ingresso un array di `double` di dimensione 3×3, e restituisca in uscita il valore del determinante della matrice. (*Suggerimenti.* Per il calcolo del determinante, potete semplicemente scrivere il codice che implementa lo sviluppo di Laplace della matrice. Si noti bene che la funzione `det` è dedicata al calcolo del determinante di matrici 3×3: non si richiede un'implementazione della funzione parametrica nella dimensione della matrice, che sarebbe notevolmente più difficile.) Usate poi la formula matriciale per il calcolo dell'area dei triangoli piani. Se i vertici del triangolo sono i tre punti $a = (x_1, y_1)$, $b = (x_2, y_2)$, $c = (x_3, y_3)$ del piano cartesiano, l'area del triangolo è data dalla metà del valore assoluto del determinante

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Per il calcolo del valore assoluto di valori `double` potete usare la funzione `double fabs(double)` del file di intestazione `math.h`, oppure implementare il calcolo direttamente.

Prima di proseguire con gli altri esercizi, è opportuno che vi assicuriate che la vostra implementazione sia corretta fino a questo punto. Scrivete una funzione `main` per testare le funzioni che avete scritto. Più in generale, cercate di testare la vostra soluzione a ciascun esercizio prima di andare avanti.

Esercizio 2

Prima versione del `main`: lettura dei dati dal file.

Punti: 8.

Scrivete una funzione `main` che accetti in ingresso dalla riga di comando il nome di un file. Se non vi sono argomenti sulla riga di comando, il programma termina con un messaggio d'errore appropriato. Se vi è almeno un argomento, il programma tenta di aprire il file dal nome corrispondente in modalità di lettura. Se l'apertura non riesce, il programma termina con un messaggio d'errore appropriato.

Se invece l'apertura avviene correttamente, il programma assume che il file sia composto esattamente nel modo seguente: la prima riga contiene un numero intero $n > 0$ seguito dal ritorno a capo. Poi, il file contiene n righe di dati, ciascuna delle quali rappresenta un triangolo dato dall'elenco dei suoi tre vertici nel formato:

`(x1,y1) (x2,y2) (x3,y3)`

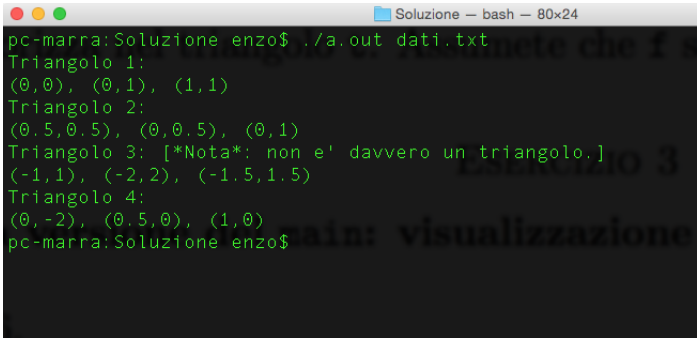

FIGURA 1. Esempio di esecuzione (Esercizio 3).

dove x_i e y_i sono valori `double`, per $i = 1, 2, 3$. Ciascuna riga è terminata dal ritorno a capo. (*Suggerimenti.* Usate `fscanf` per leggere i valori dal file. Assicuratevi di trattare i ritorni a capo che terminano ciascuna riga del file in modo opportuno.)

Il programma dichiara un array di triangoli della dimensione appropriata, e lo inizializza leggendo i dati relativi agli n triangoli dal file. Fatto ciò, il programma visualizza l'elenco dei triangoli letti, e termina. Il programma deve chiudere il file non appena non è più necessario tenerlo aperto.

Nell'implementare il programma, dichiarate, implementate e usate opportunamente la funzione ausiliaria

```
legge_tr(T* t, FILE* f)
```

che legge dal file `f` passato in argomento i dati corrispondenti a un singolo triangolo, e li memorizza nel triangolo `t`. Assumete che `f` sia già aperto in lettura.

ESERCIZIO 3

Seconda versione del `main`: visualizzazione dei dati.

Punti: 6.

Ampliate la vostra prima versione della funzione `main` (Esercizio 2) di modo che il programma, prima di terminare, visualizzi sul terminale l'elenco dei triangoli letti da file, in un formato simile a quello della Fig. 1. Dovrete dichiarare, implementare e usare opportunamente le due funzioni ausiliarie seguenti.

- `void visualizza_ar(int n, T* tr)` — visualizza l'array di triangoli di dimensione n in argomento, utilizzando `visualizza`. Se il triangolo i-esimo non è un vero triangolo (vedi sotto per i dettagli), aggiunge una nota per avvisare l'utente.
- `visualizza` — visualizza il triangolo passato in argomento. Completate opportunamente il prototipo della funzione.

Può accadere che tre punti $a = (x_1, y_1)$, $b = (x_2, y_2)$, $c = (x_3, y_3)$ del piano cartesiano non possano essere i vertici di un triangolo. Ciò avviene esattamente quando i punti non sono affinemente indipendenti, ossia quando sono allineati. Equivalentemente, ciò avviene esattamente quando l'area del "triangolo" in questione, calcolata

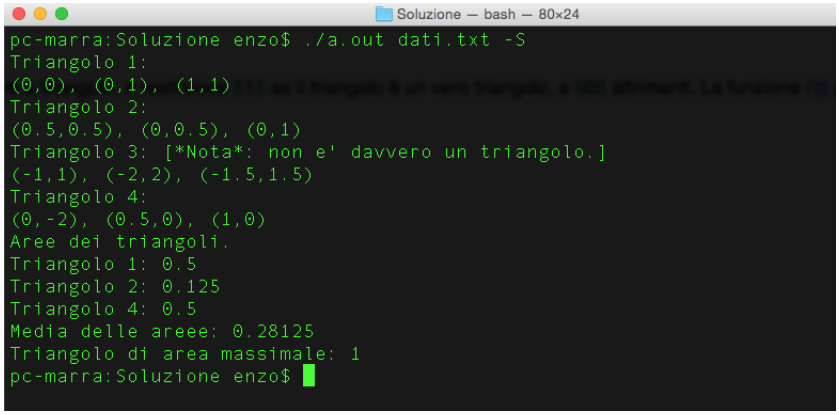

FIGURA 2. Esempio di esecuzione (Esercizio 4).

come nell'Esercizio 1, è zero. Per accertarsi del fatto che il triangolo corrente sia davvero un triangolo, la funzione `visualizza_ar` usa una funzione ausiliaria

```
affind
```

di cui dovrete completare opportunamente il prototipo, che accetta in ingresso un puntatore a triangolo, e restituisce 1 se il triangolo è un vero triangolo, e 0 altrimenti. La funzione `affind`, a sua volta, utilizza la funzione scritta nell'Esercizio 1 per il calcolo dell'area di un triangolo.

ESERCIZIO 4

Ultima versione del `main`: statistiche.

Punti: 8.

Modificate la funzione `main` scritta nell'Esercizio 3 di modo che il programma accetti una seconda opzione dalla riga di comando, oltre al nome del file in ingresso. Dopo aver visualizzato i dati come nell'Esercizio 3, il programma analizza l'eventuale secondo parametro presente sulla riga di comando. Se non vi è un secondo parametro presente, il programma termina. Se vi è un secondo parametro ed esso comincia con la stringa `-s` o `-S`, il programma computa e visualizza alcune semplici statistiche come specificato di seguito, e termina. Altrimenti, il programma termina.

Le statistiche da calcolare e visualizzare sono l'elenco delle aree dei triangoli precedentemente letti dal file, la media aritmetica delle aree e il numero di un triangolo di area massimale. L'elenco delle aree deve omettere i triangoli di area nulla. La media aritmetica include invece anche i valori nulli.

Per calcolare questi dati dichiarate e implementate due funzioni ausiliarie. La prima, di prototipo

```
double media(double *, int)
```

riceve in ingresso un array di `double` come primo parametro, la sua dimensione come secondo parametro, e restituisce la media aritmetica dei valori dell'array. La

seconda, di nome

`max`

e prototipo appropriato, calcola l'indice di un elemento di valore massimale di un array di `double`. La Fig. 2 riporta un esempio di esecuzione del programma.

Istruzioni per la consegna

- Funzioni da consegnare (in un unico file sorgente).

  `main()`
  `det()`
  `area()`
  `affind()`
  `legge_tr()`
  `visualizza()`
  `visualizza_ar()`
  `media()`
  `max()`
  Tutte le eventuali altre funzioni ausiliarie
  che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
E-mail address: `vincenzo.marra@unimi.it`