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

2016–2017

17.VII.2017

VINCENZO MARRA

INDICE

Avvertenza. Il tema d'esame richiede lo sviluppo di un singolo programma che permetta all'utente di giocare ad una variante del gioco Battaglia Navale in cui le navi sono triangoli nel piano cartesiano. Il programma leggerà i triangoli da un file specificato dall'utente sulla riga di comando. Per testare il programma userete i due file `gioco1.txt` e `gioco2.txt` allegati al testo e descritti più avanti. All'inizio del gioco il programma elenca i triangoli letti da file, e fornisce all'utente la distanza media fra le coppie di triangoli presenti nel file, assieme al massimo fra i diametri dei triangoli. (Le definizioni e i dettagli sono dati sotto.) L'utente spara un colpo dando le coordinate di un punto del piano. Se il punto giace all'interno di un triangolo, il triangolo è colpito e affondato. Assumerete sempre che i triangoli siano non degeneri (ossia che i loro vertici siano

Ultima revisione: 17 luglio 2017.

affinemente indipendenti) e disgiunti a coppie. Svilupperete delle funzioni ausiliarie nel risolvere gli esercizi. Integrerete poi tutte le funzioni ausiliarie in un unico programma finale, che sarà costituito da un solo file sorgente. (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 con attenzione prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale.

Esercizio 1

Strutture di dati, funzioni `diam`, `dist` e `centr`.

Punti: 7.

Definite con `struct` e `typedef` due nuovi tipi di dati chiamati `punto` e `tri` i cui valori permettano di rappresentare, rispettivamente, punti e triangoli nel piano cartesiano. I punti sono rappresentati da una coppia di valori `double`, ossia dalle loro coordinate. I triangoli sono rappresentati dai loro tre vertici, che sono punti nel piano. Dichiarate e implementate una funzione di prototipo

```
double diam(tri *)
```

che restituisca il diametro del triangolo ricevuto come parametro. Si ricorda che il diametro di un triangolo è la lunghezza del suo lato più lungo. Per calcolare le distanze nel piano cartesiano è necessario estrarre la radice quadrata: a tal fine potrete usare la funzione `double sqrt(double)` della libreria standard, dichiarata nel file di intestazione `math.h`. Troverete utile, anche in vista degli esercizi successivi, l'implementazione di una funzione ausiliaria che calcoli la distanza di due punti ricevuti come parametri in ingresso. Tale funzione avrà come prototipo

```
double dist(...)
```

dove dovrete fare una scelta appropriata per i parametri ricevuti in ingresso dalla funzione.

Dichiarate e implementate poi una funzione di prototipo

```
punto centr(tri *)
```

che restituisca il centroide del triangolo ricevuto come parametro. Si ricorda che dato un triangolo nel piano di vertici $v = (x_1, y_1)$, $w = (x_2, y_2)$ e $z = (x_3, y_3)$, il suo *centroide* (o *baricentro*) è il punto

$$\left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}\right).$$

Si consiglia di testare l'implementazione di `diam`, `dist` e `centr` prima di proseguire. A tal fine potrete scrivere una breve procedura `main` che invochi le funzioni con parametri appropriati e visualizzi il valore che essa restituisce.

Esercizio 2

Calcolo del diametro massimo e della distanza media.

Punti: 7.

Aggiungete adesso al vostro programma una variabile globale di nome

`elenco`

e di tipo appropriato, atta a rappresentare un array di triangoli che sarà allocato dinamicamente più avanti.

Implementate una funzione di prototipo

```
double diammax(int n)
```

che assuma che la variabile `elenco` rappresenti un array di `n` triangoli, e restituisca il valore del diametro massimo fra quelli dei triangoli nell'elenco. Se il parametro in ingresso è negativo o nullo la funzione restituisce, per convenzione, `-1`.

Implementate poi una funzione di prototipo

```
double distmed(int n)
```

che assuma che la variabile `elenco` rappresenti un array di `n` triangoli, e restituisca il valor medio delle distanze fra le coppie di triangoli presenti nell'array. Per *distanza* fra due triangoli s'intende, ai fini di questo tema d'esame, la distanza euclidea fra i loro centroidi. Se il parametro in ingresso è negativo o nullo la funzione restituisce, per convenzione, `-1`.

Esercizio 3

Lettura dei dati da file e allocazione della memoria.

Punti: 6.

Scrivete una funzione di prototipo

```
int caricatri(char* nomef, int* n)
```

che legga dal file il cui nome è la stringa `nomef` un elenco di triangoli, e allochi la memoria necessaria per memorizzarli. Dopo l'esecuzione della funzione, l'intero `*n` dovrà essere pari al numero di triangoli contenuti nel file, di modo che il chiamante possa sfruttare questa informazione.

La funzione assume che il formato del file il cui nome è passato in argomento sia esattamente quello dei due file d'esempio allegati al testo. In dettaglio, ciascun file è costituito da una prima riga contente il numero (intero positivo) `n` di triangoli codificati nel file. Questa riga è seguita da `n` righe successive, ciascuna contente la codifica di un triangolo nella forma:

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

(Non vi sono spazi fra i caratteri della riga.) Le Figg. 1 e 2 mostrano la rappresentazione grafica dei triangoli codificati nei file allegati `gioco1.txt` e `gioco2.txt`.

La funzione `caricatri` tenta di aprire il file `nomef` in lettura. Se la stringa `nomef` è `NULL` o si rileva un errore nell'apertura del file, la funzione termina restituendo `-1` al chiamante. Se invece non vi sono errori, la funzione legge innanzitutto il numero di triangoli dalla prima riga, memorizzandone il valore in `*n`. Poi la funzione alloca la variabile globale `elenco` di modo che essa rappresenti un array di `n` triangoli. Se si riscontra un errore nell'allocazione della memoria, la funzione termina restituendo `-2` al chiamante. Altrimenti, la funzione legge i triangoli dalle `n` righe successive

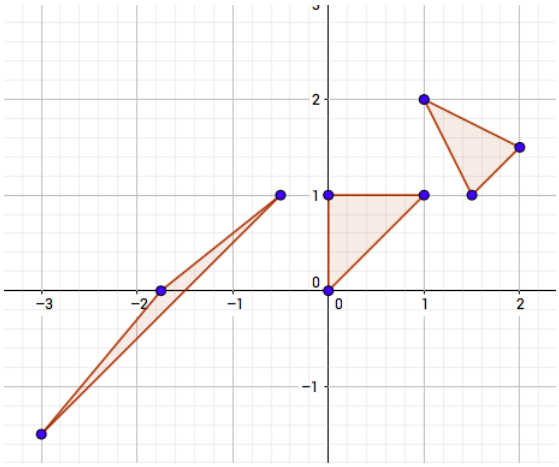

FIGURA 1. I triangoli codificati dal file `gioco1.txt`.

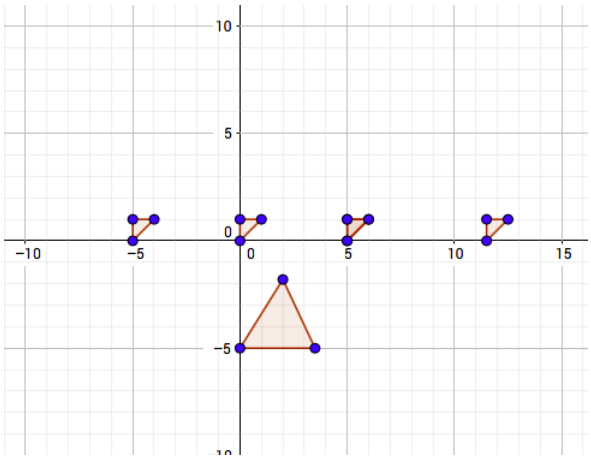

FIGURA 2. I triangoli codificati dal file `gioco2.txt`.

del file i triangoli e li memorizza nell'array `elenco`. Infine, la funzione restituisce il controllo al chiamante. Assicuratevi di aver chiuso il file nei punti opportuni: dopo l'esecuzione della funzione il file deve risultare chiuso.

ESERCIZIO 4

Funzioni ausiliarie `app`, `leggi` e `vis`.

Punti: 5.

Scrivete tre ultime funzioni ausiliarie prima di mettere assieme il tutto in un programma funzionante.

- La funzione `void vis(int n)` visualizza i triangoli contenuti in `elenco`, che è assunto avere lunghezza `n`, in un formato simile a quello delle Figg. 3 e 4.
- La funzione `punto* leggi(void)` legge da tastiera le coordinate di un punto inserito dall'utente, nel formato `(x,y)` seguito da invio, alloca tramite `malloc` la memoria necessaria a memorizzare il punto, e restituisce al chiamante un puntatore al punto. Se riscontra un errore nell'allocazione della memoria oppure un errore nel formato dei dati inseriti dall'utente, restituisce `NULL`. (Questa funzione è usata, negli esempi delle Figg. 3 e 4, per leggere ciò che digita l'utente dopo la scritta `Spara:`.)
- La funzione `int app(tri *t, punto *p)` stabilisce se il punto passato come secondo parametro appartiene al triangolo passato come primo parametro. Se uno fra i puntatori ricevuti in ingresso è `NULL`, restituisce `0`. Altrimenti, la funzione calcola le tre *coordinate baricentriche* (cfr. oltre) (a, b, c) del punto `p` rispetto al triangolo `t`. Poi controlla se le condizioni
$$0 \le a \le 1,$$
$$0 \le b \le 1,$$
$$0 \le c \le 1$$
sono soddisfatte. In questo caso, e solo in questo caso, il punto `p` appartiene a `t`, e la funzione restituisce il valore `1`. Altrimenti, la funzione restituisce il valore `0`.

Per calcolare le coordinate baricentriche (a, b, c) di un punto $p = (x, y)$ rispetto a un triangolo di vertici $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, si usino le formule
$$a = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$
$$b = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$
$$c = 1 - a - b.$$

```
Vincenzos-MacBook-Air:Soluzione enzo$ ./a.out gioco1.txt
Navi caricate.
T1:      (0,0),(0,1),(1,1)
T2:      (-0.5,1),(-1.75,0),(-3,-1.5)
T3:      (1.5,1),(2,1.5),(1,2)
Diametro massimo delle navi: 3.53553
Distanza media fra le navi: 2.44332
Spara: (0.5,0.5)
Hai colpito e affondato la nave T1!
Rimangono 2 navi.
Spara: (0.5,0.52)
Avevi gia' affondato la nave T1, peccato.
Spara: (1.5,-2)
Hai mancato il bersaglio!
Spara: (1.5,1)
Hai colpito e affondato la nave T3!
Rimane 1 nave.
Spara: (-0.5,0.99)
Hai mancato il bersaglio!
Spara: (-0.6,0.90)
Hai colpito e affondato la nave T2!
Rimangono 0 navi.
Hai affondato tutte le navi! Fine del gioco.
Vincenzos-MacBook-Air:Soluzione enzo$
```

FIGURA 3. Esempio di esecuzione del programma completo.

```
Vincenzos-MacBook-Air:Soluzione enzo$ ./a.out gioco2.txt
Navi caricate.
T1:      (0,0),(0,1),(1,1)
T2:      (5,0),(5,1),(6,1)
T3:      (-5,0),(-5,1),(-4,1)
T4:      (11.5,0),(11.5,1),(12.5,1)
T5:      (0,-5),(3.5,-5),(2,-1.8)
Diametro massimo delle navi: 3.77359
Distanza media fra le navi: 8.40888
Spara: (0,12)
Hai mancato il bersaglio!
Spara: (0,1)
Hai colpito e affondato la nave T1!
Rimangono 4 navi.
Spara: (3.5,-4.99)
Hai mancato il bersaglio!
Spara: (ke
Formato non valido o errore nell'allocazione della memoria. Riprova.
Spara: (3.5,-3)
Hai mancato il bersaglio!
Spara:
```

FIGURA 4. Esempio di esecuzione del programma completo.

ESERCIZIO 5

La procedura `main`.

Punti: 5.

Scrivete una procedura `main` che metta insieme in un unico programma le funzioni sviluppate fin qui. Il programma deve comportarsi come segue.

- L'utente specifica sulla riga di comando il nome del file nei quali sono codificati i triangoli. Se non vi sono argomenti sulla riga di comando il programma informa l'utente e termina.
- Il programma carica i triangoli da file e alloca la memoria. Se vi sono errori nell'apertura del file o nell'allocazione, il programma informa l'utente e termina.
- Il programma visualizza l'elenco dei triangoli, il loro diametro massimo la media delle distanze fra le coppie di triangoli. Si vedano le Figg. 3 e 4.
- Il programma avvia quindi il gioco chiedendo all'utente di sparare un colpo, ossia di specificare un punto in coordinate cartesiane.
- Se il punto specificato dall'utente appartiene a un (e quindi a un solo) triangolo, il triangolo è colpito e affondato. Altrimenti, il bersaglio è mancato. Usate un appropriato array di interi per tenere traccia di quali triangoli siano già stati affondati.
- Il gioco e il programma terminano quando i triangoli sono stati tutti colpiti.

Istruzioni per la consegna

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

  `main`

  `diam`

  `dist`

  `centr`

  `distmed`

  `diammax`

  `caricatri`

  `vis`

  `leggi`

  `app`

  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