INFORMATICA (ING. BIOMEDICA) **Scritto 8 settembre 2011**

NOTE SULLO SVOLGIMENTO DELLA PROVA:

- PER SVOLGERE L'ELABORATO, APRIRE il Dev-C++ (dal Menù **Avvio** (o **Start**) nella barra degli strumenti in fondo allo schermo, selezionare Programmi e quindi Dev-C++);
- PRIMA DI INIZIARE LO SVOLGIMENTO DELL'ELABORATO, selezionare la voce **Identifica studente** nel menù **Strumenti** all'interno dell'ambiente Dev-C++ e inserire i dati richiesti;
- per svolgere l'elaborato, aprire il progetto *APRIQUESTO.dev* presente nel directory *c:\esame\esaInf* e scrivere le funzioni richieste nel file *compito.cpp*, già presente nel progetto;
- per una corretta stampa dell'elaborato bisogna mantenere il codice entro i margini imposti dall'ambiente Dev-C++ (linea verticale presente alla destra della pagina).

Percorsi

Supponiamo che la posizione di un corridore sia acquisita attraverso un opportuno apparecchio, per esempio un ricevitore GPS. Supponiamo inoltre che, per semplicità, il movimento dei corridori avvenga in un piano cartesiano. Un programma per la gestione dei percorsi seguiti dai corridori può essere realizzato usando le seguenti costanti e i seguenti tipi:

```
const int MAXT = 50;
const int MAXS = 100;
const int MAXP = 100;
const int MAXN = 80;
struct Tratto {
  float startx;
  float starty;
  float endx;
  float endy;
  float tempo;
};
struct Percorso {
  char nome[MAXS];
  Tratto tr[MAXT];
  int numTr;
};
struct ArchivioPercorsi {
  Percorso per[MAXP];
  int numPer;
};
```

Il percorso seguito da un atleta durante un allenamento può essere schematizzato come una spezzata in cui, per ogni tratto della spezzata, la velocità si mantiene costante. Ogni tratto della spezzata (un segmento) è caratterizzato dalle coordinate di partenza (`startx` e `starty`), dalle coordinate di arrivo (`endx` e `endy`), e dal tempo impiegato a percorrerlo (`tempo`). La struttura `Percorso` memorizza il nome dell'atleta (campo `nome`) e i dati del percorso. Quest'ultimo viene memorizzato come un insieme di tratti nell'array `tr` (al più `MAXT` elementi), mentre il campo `numTr` indica il numero di tratti che compongono lo specifico percorso. La struttuna `ArchivioPercorsi` memorizza i percorsi nell 'array `per` (il numero di percorsi realmente presenti è indicato da `numPer`). Nell'archivio possono essere contenuti più percorsi aventi lo stesso nome di atleta.

Scrivere il corpo delle seguenti funzioni C++.

1. **`int carica(ArchivioPercorsi* pa, const char nomef[])`** che riempie l'archivio con i dati contenuti nel file di nome `nomef`. Il file contiene i dati dei percorsi nel formato indicato dal seguente esempio:

   Francesco
   3
   23.0 24.0 26.0 25.0 14.5
   26.0 25.0 31.1 25.5 8.5
   31.1 25.5 31.4 28.7 11.5

Alfredo
4
13.0 14.0 16.0 15.0 12.5
16.0 15.0 33.3 22.2 8.4
33.3 22.2 33.6 24.5 10.1
33.6 24.5 31.0 19.5 11.0

...

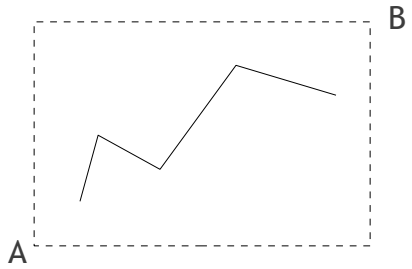
Per ogni percorso abbiamo: una stringa che rappresenta il nome dell'atleta, un intero che indica il numero di tratti di cui si compone lo specifico percorso e un numero di quintuple pari al numero di tratti. Ogni quintupla contiene valori reali corrispondenti, nell'ordine, a: `startx`, `starty`, `endx`, `endy`, `tempo`. La funzione restituisce un valore che indica quanti percorsi sono stati letti.

2. **`float distanzaTotale(const ArchivioPercorsi* pa, const char n[])`** che restituisce la distanza totale percorsa dall'atleta di nome `n` (tutti i percorsi in archivio relativi all'atleta `n`).

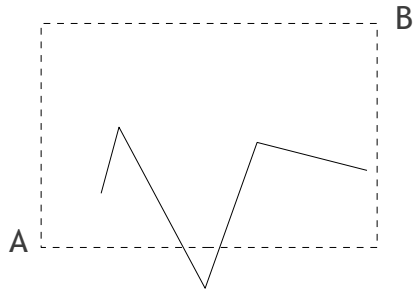
3. **`void stampaNomiOrd(const ArchivioPercorsi* pa)`** che stampa a video i nomi dei diversi atleti presenti nell'archivio e il numero di percorsi fatti da ogni atleta. La lista deve essere ordinata per valori decrescenti del numero di percorsi, come illustrato dal seguente esempio:

Mario	*3*
Giuseppe	*2*
Carlo	*1*
Francesco	*1*
Alfredo	*1*

4. **`int seleziona(const ArchivioPercorsi* pa, float ax, float ay, float bx, float by, Percorso v[])`** che riempie l'array `v` con i percorsi che sono interamente contenuti nel rettangolo identificato da due punti A e B, come illustrato dal seguente esempio:



Interamente contenuto



Non interamente contenuto

I parametri `ax` e `ay` rappresentano le coordinate x e y di A, mentre `bx` e `by` rappresentano le coordinate x e y di B. La funzione restituisce il numero di elementi inseriti in `v`.

5. **`void disegnaGrafico(const Percorso* pp)`** che stampa a video l'andamento delle velocità nei vari tratti che compongono un percorso. Ogni colonna corrisponde a un tratto e in ogni colonna sono presenti uno o due asterischi, a seconda della velocità del tratto. In particolare, sono presenti
   - due asterischi se la velocità è maggiore della velocità media del percorso;
   - un asterisco se la velocità è minore della velocità media del percorso.

Esempio relativo a un percorso con sei tratti:

```
Eleonora

 *** *
******
```