Scheda di lavoro N. 11

Questa è la *scheda delle strutture*. Si tratta, per l'appunto, di esercizi che richiedono la definizione e l'uso di strutture. Il primo esercizio è elementare, e mi aspetto che tu riesca a farlo senza eccessive difficoltà. Il secondo esercizio aggiunge solo la difficoltà della lettura da file, e non dovrebbe rivelarsi particolarmente complicato.

1. Il linguaggio C non prevede (non ancora, almeno) i numeri complessi. Occorre organizzarsi. Il ricorso alle strutture – sia pure a livello davvero elementare – può rivelarsi molto utile, in un caso come questo. L'esercizio che ti propongo è costruirti un insieme di funzioni che eseguano le operazioni elementari sui numeri complessi.

Suggerimenti:

- inizia col preparare un file `complessi.h` che contenga la definizione del tipo `COMPLESSO` come struttura. In seguito, aggiungerai a questo file i prototipi delle funzioni che hai scritto. Trovi un esempio sul dischetto.
- Scrivi delle funzioni elementari

```
double modulo(COMPLESSO a);
double argomento(COMPLESSO a);
COMPLESSO coniugato(COMPLESSO a);
COMPLESSO somma(COMPLESSO a,COMPLESSO b);
COMPLESSO prodotto(COMPLESSO a,COMPLESSO b);
COMPLESSO rapporto(COMPLESSO a,COMPLESSO b);
COMPLESSO inverso(COMPLESSO a);
```

  che ti mettano in grado di calcolare modulo e argomento, e di eseguire le operazioni aritmetiche fondamentali sui numeri complessi. Potrai mettere tutte queste funzioni in un unico file `complessi.c`. Non dimenticare di inserire un

```
#include "complessi.h"
```

  Troverai un esempio sul dischetto.
- Scrivi il solito programmino di prova che esegua tutte le operazioni e stampi i risultati. Anche per questo troverai un esempio nel file `prova_complessi.c` sul dischetto. Troverai anche le istruzioni per la compilazione nel file `Makefile`.

A questo punto potrai considerare finito l'esercizio. Se ne avrai bisogno, non avrai difficoltà in futuro a completare il pacchetto di funzioni inserendo anche il calcolo della radice quadrata, di una potenza, delle funzioni esponenziali e trigonometriche o altro.

2. Il file `aster.dat` che trovi su dischetto contiene un catalogo di elementi orbitali di un gruppo dei asteroidi (detti *Trojani*). Sulla prima riga trovi il significato delle colonne. Ecco qualche spiegazione aggiuntiva.

- Le prime due colonne contengono il numero di catalogo ed il nome (se sono già stati assegnati).
- `Ln` identifica il punto Lagrangiano in prossimità del quale si trovano gli asteroidi; i due punti interessati si chiamano L_4 ed L_5.
- `q` e `Q` sono le distanze dal sole del perielio e dell'afelio, rispettivamente.

- `Epoch` è l' epoca a cui si riferiscono gli elementi orbitali che seguono, espressa come anno (4 caratteri) giorno (2 caratteri) e mese (2 caratteri). Per la maggior parte dei casi il dato si riferisce al 4 gennaio 2001.
- `M` è l' anomalia media;
- `Peri` è la longitudine del perielio
- `Node` è la longitudine del nodo ascendente;
- `Incl` è l' inclinazione del piano dell' orbita;
- `e` è l' eccentricità dell' orbita;
- `a` è il semiasse maggiore dell' orbita.

Se hai capito poco o nulla dei termini tecnici usati qui sopra, non angosciarti: capire esattamente cosa siano queste informazioni non è molto importante al fine di svolgere l' esercizio proposto. Il problema qui è saper scrivere un programma che legga il file e poi faccia qualche elaborazione (più o meno complessa) sui dati. Prova a scrivere un programma che:

i. definisca una struttura che contenga tutti i dati relativi ad un asteroide, ed allochi memoria per un numero sufficientemente alto di tali strutture;
ii. legga il file;
iii. calcoli la media di alcune quantità, ad esempio inclinazioni, eccentricità e semiassi maggiori.
   Suggerimenti:

- Anche qui, ti conviene spezzare il programma in sottoprogrammi brevi: uno per la lettura dei dati, ed altri per il calcolo delle medie.
- La parte complicata del programma è la lettura del file. In particolare, leggere la parte non numerica (il numero di catalogo, che è messo tra parentesi, il nome e la colonna `Ln`) può mettere a dura prova la tua pazienza, perchè il sottoprogramma `fscanf()` ha un comportamento un po' bizzarro sulla lettura di caratteri. Eccoti alcune indicazioni.
- Per eliminare le prime due righe (che non contengono dati) ti conviene usare la funzione di libreria `fgets`. La sintassi è:
  `fgets(riga,n,file)`
  dove `riga` è l' indirizzo di una stringa di caratteri dove verrà memorizzata la riga letta; `n` è il numero massimo di caratteri nella riga; `file` è il puntatore al file da cui si deve effetture la lettura. Quando non trova più nulla da leggere, `fgets` restituisce il valore `NULL`.
- Per superare il problema di come saltare i primi 24 caratteri (dall' inizio della riga fino a 'L' di `L4` o `L5`) puoi procedere così:
  - definisci una stringa di caratteri `riga` sufficiente a contenere la riga intera;
  - usando la funzione `fgets` leggi l' intera riga di dati;
  - converti i dati usando la funzione
    `sscanf(riga+24,''`<*formato*>`'',`<*indirizzi dei dati*>`);`

  funziona come `scanf`, ma prende i dati in input a partire dalla 24–esima posizione della stringa `riga` invece che da terminale.
- Se vuoi saperne di più sulle funzioni `fgets` e `sscanf`, basta leggere le pagine del manuale.

- Se dopo due giorni sei ancora in difficoltà con la lettura, trovi un esempio nel programma `asteroid.c`, sul dischetto.