Linguaggi I2

Serie files e memoria dinamica

1. Modificare il modulo stack realizzato staticamente (serie 2b, esercizio in classe), in modo che faccia uso di:

- array dinamici al suo interno (variante 1), incrementando (e decrementando) di un valore di offset la dimensione, a seconda della richiesta

- liste semplici concatenate (variante 2), considerando di far crescere e decrescere la lista ad ogni modifica.

Utilizzare un debugger e verificare il funzionamento passo per passo.

3. Data la struttura

```
struct libro {
  char titolo[40];
  char autori[30];
  int anno;
  char isbn[10];
  int pagine;
};

typedef struct libro libro;
```

scrivere le funzioni seguenti e un programma principale che ne faccia uso:

a) `int leggi_file(char *nome_file, lib *libri, int lun_libri)`
Legge i dati contenuti nel file di nome "nome_file" nell'array libri che ha una lunghezza massima di "lun_libri". La funzione restituisce il numero di libri letti.
Supponiamo in questo caso che la lunghezza dell'array sia maggiore del numero di elementi contenuti nel file.

c) `int scrivi_file(char *nome_file, lib *libri, int lun_libri)`
Scrive i dati contenuti nell'array libri, di lunghezza "lun_libri", su un file di nome "nome_file". La funzione restituisce il numero di libri scritti.

d) `int trova_libri(char *inizio_tit,lib *libri,int lun_libri,lib *risultato)`
Inserisce nell'array "risultato" (che può essere lungo almeno tanto quanto "libri") tutti i libri dell'array "libri" che abbiano il titolo che inizia con la stringa "inizio_tit" (o che siano uguali). La funzione restituisce il numero di libri trovati.

3. Si abbia un file binario "impiegati", contenente tra le altre cose nome, cognome, indirizzo e stipendio. Siano escluse omonimie di nome e cognome.

Si abbia poi un file di testo "aggiornastipendi", editabile, contenente nome, cognome e nuovo valore di stipendio di alcuni impiegati.

Scrivere il codice che permetta di aggiornare il file "impiegati" partendo dal file "aggiornastipendi".

I file non sono ordinati. Trovare vari procedimenti.

4. Scrivere la funzione

```
float *leggi_note(FILE *fp, int &numNote)
```

simile a quella realizzata nell'esercizio 1.
In questo caso la funzione alloca al suo interno un array dinamico e legga le note da un file. Il numero di elementi letti (corrispondente alla dimensione del file), viene restituito attraverso il parametro "per indirizzo", mentre l'indirizzo del nuovo array creato **dinamicamente** viene restituito come valore di return.

Scrivere un programma di test per l'utilizzo di questa funzione.

5. Modificare il modulo stack realizzato staticamente (serie 2b, esercizio in classe), in modo che faccia uso di array dinamici al suo interno.