# Laurea In Ingegneria dell'Informazione

# Esercitazioni Guidate di Tecniche della Programmazione

**Note introduttive: Seguire le raccomandazioni date nelle precedenti EG …**

## 8. Esercitazione 8 – file binari

### *8.1. File binari (BINARI1.C)*

Scrivere un programma che legga, oppure assegni in diverso modo, i valori di un array di double `arrayDouble` e poi memorizzi i valori i valori di `arrayDouble` in un file binario.
Il programma deve leggere da input il nome del file in cui memorizzare i dati. Il programma deve usare la funzione `fwrite()` per memorizzare i dati nel file; in particolare **è richiesto** che il terzo parametro delle chiamate di `fwrite()` sia, in questo esercizio, sempre uguale ad 1.

### *8.2. File binari (BINARI2.C)*

Scrivere un programma che usa una variabile double `val` e un array di double `arrayDouble` per contenere certi valori double. I valori contenuti in `val` e `arrayDouble` devono essere letti da un file binario (quello costruito nell'esercizio precedente), usando la funzione `fread()`. **E` richiesto** che il terzo parametro delle chiamate di `fread()` sia, in questo esercizio, sempre uguale ad 1. In particolare, il programma dovrebbe memorizzare in val il primo dei valori letti dal file, mentre i successivi valori presenti nel file dovrebbero finire nell'array.

### *8.3. File binari (BINARI3.C, BINARI4.C)*

Ripetere gli esercizi dei due punti precedenti. Il limite imposto precedentemente sul terzo parametro delle funzioni `fread()` ed `fwrite()`e` cancellato ...

### *8.4. File binari (BINARI5.C)*

Scrivere un programma che
- chieda e legga un numero intero `n`
- e legga poi `n` numeri double
- memorizzando tutti i dati letti (interi e double) in un file binario il cui nome e` stato anche letto da input (si suggerisce di farlo prima di leggere i dati numerici)

Il file prodotto conterra`, come primo dato il valore `n` e conterra` successivamente tutti i double letti. Non serve usare array!

### *8.5. File binari (BINARI6.C)*

Scrivere un programma che

- chieda e legga il nome di un file binario, analogo a quelli prodotto con il programma dell'esercizio precedente)
- legga dal file un numero intero n
- e legga i successivi n numeri double, memorizzandoli in un array dinamico di n double (cioe` allocato "esattamente")

### *8.6. Quadrilateri e file binari (QUADRI2.C)*

Scrivere un programma capace di leggere una sequenza di quadrilateri da standard input (stdin, la tastiera) memorizzandoli in un file binario.
Il programma deve chiedere il nome del file binario in cui effettuare le memorizzazioni e poi deve leggere quadrilateri e memorizzarli fintantoche` l'utente intende continuare a fornire dati.

### *8.7. Quadrilateri e file binari (QUADRI3.C)*

Scrivere un programma capace di leggere da un file costruito dal programma precedente.
Il programma deve chiedere il nome del file di quadrilateri da usare come file di input ed utilizzarlo per leggere i quadrilateri in esso contenuti. Ogni quadrilatero deve essere stampato in standard output (sul video).

Più giù c'è un suggerimento: non guardarlo. (Non guardarlo subito …)

**Suggerimento:**

ridefinire le funzioni di

- lettura di un punto (funzione che riceve l'indirizzo di un punto e una variabile file, gia` legata ad un file binario, e legge dal file unpunto con fread());
- e di lettura di un quadrilatero (funzione che riceve l'indirizzo di un quadrilatero e una variabile file, gia` legata ad un file binario, e legge dal file i quattro vertici del quadrilatero, usando la funzione di lettura punti da file).

### *8.8. Quadrilateri e file binari (QUADRI4.C)*

Scrivere un programma capace di

- ricevere il nome, `n1`, di un file binario, contenente quadrilateri; (si suppone che questi quadrilateri siano non degeneri, cioe` aventi, quali vertici, quattro punti distinti)
- produrre un nuovo file (sempre binario, e il cui nome, `n2`, e` stato fornito in input) contenente i quadrilateri registrati nel file `n1` che siano quadrati.

Più giù qualche suggerimento

**Suggerimento:**
`n1` viene usato come file di input, aperto in lettura; `n2` sara` il file di output, aperto in scrittura.
Si legge un quadrilatero da n1, si controlla e , se e` un quadrato, lo si registra in n2.
Dato che i due file rimangono aperti contemporaneamente, servono due variabili FILE da collegare ad essi.

Seguono altri suggerimenti …

**Suggerimento 2 di 5:**

Per controllare se un quadrilatero e` un quadrato sarebbe elegante usare una funzione `isQuadrato()` che, ricevendo un quadrilatero, restituisca 1 se si tratta di un quadrato e 0 altrimenti.

Suggerimento segue

**Suggerimento 3 di 5:**

Un'ida della parte di algoritmo che si occupa di estrarre dati dal primo file e (decidere se) riversarli nel secondo:

```
mentre il file di input contiene dati
          - si legge un quadrilatero
          - si controlla se e` quadrato e,
            se e` quadrato, si registra nel file di output
```

Suggerimento segue

**Suggerimento 4 di 5:**

Supponendo che quad sia il quadrilatero e f2 la variabile file associata al file di output, l'istruzione di registrazione del quadrilatero nel file potrebbe essere:

```
fwrite(quad, sizeof(TipoPunto), 4, f2)
```

Suggerimento segue

**Suggerimento 5:**

Tra i vari controlli da inserire nel programma, ci sono :

- controllo se il file di input e` stato aperto regolarmente;
- controllo se il file di output e` stato aperto regolarmente;
- controllo se la registrazione di un quadrilatero nel file di output e` stata effettuata con successo (ricordarsi che una chiamata ad `fwrite()` restituisce il numero di registrazioni che sono state effettuate con successo, quindi se la chiamata doveva registrare 4 punti e lo ha fatto con successo, restituisce 4.