Laboratorio di Calcolo per Fisici, Settima Esercitazione

Canale L-PO, Docente: Cristiano De Michele

Lo scopo della settima esercitazione di laboratorio è di fare pratica con le istruzioni di input/output da file e la gestione degli array, scrivendo un programma che simula la gestione del personale di un piccolo negozio.

► **Prima parte:**
Vediamo anzitutto come in C si possa leggere il contenuto di un file o scrivere su di esso. Come vedremo sarà necessario fare quanto segue:

1. Dichiarare un puntare di tipo FILE *.
2. Aprire il file nella modalità opportuna (ovvero in lettura o scrittura).
3. Leggere dal file o scrivere su di esso a seconda della modalità con cui lo si è aperto.
4. Chiudere il file una volta terminate le operazioni di lettura o scrittura.

Per dichiarare un puntatore di tipo file si può usare la seguente istruzione in C:

```
FILE *fpr;
```

Fatto questo il file va *aperto* tramite l'istruzione fopen(...), che richiede come argomenti una stringa con il nome del file da aprire ed una seconda stringa con cui si specifica la modalità di apertura del file (ad es. "r" per la lettura e "w" per la scrittura). Se, ad esempio, si vuole aprire un file chiamato "trimestre1.dat" in lettura si deve usare la seguente istruzione:

```
fpr=fopen("trimestre1.dat", "r");
```

Se l'apertura del file non va a buon fine, ad esempio se il file non esiste o non può essere letto, al puntatore fpr viene assegnato il valore NULL, che va inteso come un indirizzo di memoria non valido. Quindi nel caso che al puntatore venga assegnato tale valore è bene che il programma termini con un messaggio d'errore come mostrato di seguito:

```
if(fpr==NULL)
  {
    printf("Errore nell'apertura del file. Esco...\n");
    exit(1);
  }
```

Nell'esempio sopra la funzione exit() con argomento 1 termina l'esecuzione del programma.
Se, invece, si vuole aprire un file in modalità scrittura, bisogna usare il simbolo "w" (write) al posto di "r" nella fopen( ). Con questa modalità, se il file non esiste viene creato e se esiste il suo contenuto viene cancellato. Anche in tale caso, se, per qualsiasi motivo, l'apertura del file dovesse fallire, ad esempio perché il file esiste già ed è protetto in scrittura oppure perché la memoria non volatile è piena,

al puntatore fpr verrebbe assegnato il valore NULL. Come nel caso precedente è quindi opportuno verificare tale evenienza ed eventualmente uscire dal programma con un messaggio d'errore, come mostrato in precedenza.
Oltre alla modalità di scrittura "w" esiste anche la modalità "a" (append) con cui un file esistente viene aperto ed il suo contenuto preservato. In tale modalità le successive scritture su file aggiungono dati a partire dalla fine dello stesso.
Se il file è stato aperto in modalità di lettura è possibile leggerne i dati contenuti usando la funzione fscanf( ). La sintassi per l'utilizzo di tale istruzione è la seguente:

```
fscanf(fpr,"%i %i %i", &a, &b, &c);
```

dove in tale caso vengono letti da una riga del file tre interi e memorizzati nelle variabili intere a, b e c (che vanno ovviamente opportunamente dichiarate nel codice prima di essere utilizzate). Per leggere un intero file costituito da più righe è sufficiente includere l'istruzione sopra in un ciclo for( ) con un numero di cicli pari al numero di righe presenti nel file.
Analogamente, se un file è stato aperto in modalità di scrittura, per scrivere su di esso si usa la funzione fprintf(fpr,"....", ...), ovvero ad esempio:

```
fprintf(fpr,"%i %i %i\n", a, b, c);
```

stampa il contenuto delle variabili intere a, b e c su una riga del file aperto in scrittura.
Terminate le operazioni di lettura o scrittura il file va necessariamente chiuso con la seguente istruzione:

```
fclose(fpr);
```

1. Creare una cartella chiamata EX7 che conterrà il materiale dell'esercitazione.
2. Creare (con un editor) un file di testo, chiamato *trimestre1.dat*, che contenga un riassunto dei dati trimestrali di vendita dei 10 commessi di un piccolo negozio di computer. Copiate nel file i seguenti dati, divisi in tre colonne: numero del commesso; euro totali incassati; ore lavorate e 10 righe (una per commesso):

   ```
   1 450 32
   2 322 14
   3 870 82
   4 765 76
   5 198 20
   6 145 10
   7 828 75
   8 932 80
   9 267 32
   10 424 18
   ```

3. Scrivere un programma leggi.c che, utilizzando opportunamente le funzioni di lettura da un file legga i dati contenuti in *trimestre1.dat*, e salvi la seconda e terza colonna del file in un array 2d di interi chiamato dati[ ][ ] costituito da 10 righe e 2 colonne. *Suggerimento:* Notate che dovete

memorizzare solo i dati della seconda e terza colonna nell'array 2d. Per far questo potete usare la seguente istruzione in C:

```
fscanf(fpr, "%i %i %i", &dumb, &(dati[i][0]), &(dati[i][1]));
```

dove dumb è una variabile intera che andrà opportunamente dichiarata, i è una variabile intera da usare come contatore in un ciclo for() all'interno del quale andrà inserita la precedente istruzione. Si fa presente che non è necessario memorizzare la prima colonna del file perché il numero del commesso corrisponde all'indice di riga della matrice dati[ ][ ] aumentato di 1.

4. Oltre ai dati letti dal file, il programma leggi.c dovrà calcolare la *media oraria* di vendita di ciascun commesso (ossia euro guadagnati su ore lavorate di ciascun commesso) e salvarla in un array di double chiamato media[ ] di lunghezza pari a 10.

5. Dopo aver chiuso il file *trimestre1.dat*, stampare su schermo, per ciascun commesso, i dati ad esso relativi nel formato
   # Commesso # Ore # Vendite in Euro # Media

► **Seconda parte:**
Copiate il programma appena creato in uno chiamato dai_voto.c. Quest'ultimo dovrà essere modificato in maniera tale che a partire dall'elaborazione dei risultati di vendita trimestrale di ciascun commesso, stili una classifica dei 10 commessi e attribuisca loro un voto. In particolare, il programma deve:

1. Stilare una classifica dei commessi in base alla media oraria di vendita.

2. Dare a ciascun commesso un voto compreso tra 1 e 10 per valutare la sua performance; il commesso con la performance migliore ottiene 10 punti, il secondo 9, e così via.

3. Stampare su schermo e su un file le seguenti informazioni per ciascun commesso:
   # Commesso # Ore # Vendite in Euro # Media # Voto

Suggerimento: per stilare una classifica dei commessi, dovete ordinare l'array delle medie con il *bubblesort*, ma dobbiamo anche sapere a fine ordinamento a quali commessi appartengono le medie ordinate. Per fare questo si può usare un'array 2d di tipo double, che chiameremo sortmedia, costituito da 10 righe e due colonne, dove nella prima colonna vanno memorizzati i numeri dei commessi (da 1 a 10) e nella seconda colonna le corrispondenti medie. Nel bubblesort il criterio per lo scambio sarà basato sulle medie, ovvero sui valori nella seconda colonna dell'array sortmedia, ma si dovranno scambiare sia gli elementi della prima colonna che della seconda, ovvero se NC sono il numero di commessi:

```
//******* >>> bubblesort <<< *******
//doppio ciclo for del bubblesort sulle righe della matrice sortmedia
for(i=0;i<NC-1;i++)
  {
    for(j=NC-1; j > i; j--)
      {
        // esamino due righe contigue della colonna 1 della matrice
```

```
        if (sortmedia[j][1] > sortmedia[j-1][1])
          {
            // Inserisci qui il codice per scambiare
            // sia gli elementi confrontati sia i numeri
            // dei commessi, in modo che questi rimangano associati
            // alle corrispondenti medie

          }
      }
}
```

Alla fine della procedura le righe dellarray sortmedia saranno ordinate a partire dal commesso con media più alta in maniera decrescente. Quindi, scorrendo le righe dell'array sortmedia con un opportuno ciclo for(), in cui si faccia uso del contatore i, possiamo assegnare al commesso il cui numero è pari a sortmedia[i][0] un voto pari a NC-i. Per memorizzare il voto, possiamo definire un array voto[] di 10 interi che in ogni casella contenga il voto associato al commesso. Ad esempio, in voto[0] dovrà esserci il voto associato al commesso 1, voto[1] conterrà il voto associato al commesso 2 e così via fino a voto[9], che conterrà il voto del commesso 10.

► **Terza parte (facoltativa/per casa)**

Modificare il programma dai_voto.c in modo che analizzi i dati per i quattro trimestri del 2016, riportati nel riquadro sottostante, che andranno copiati in un file chiamato anno2016.dat. Il file contiene 9 colonne dove la colonna 0 rappresenta il numero del commesso e le altre colonne sono gli euro totali incassati e le ore lavorate nei quattro trimestri. In particolare: le colonne 1 e 2 si riferiscono al primo trimestre, le colonne 3 e 4 al secondo, le colonne 5 e 6 al terzo e le colonne 7 e 8 al quarto trimestre. Una volta ottenute le medie orarie e i voti di ciascun venditore per tutti i trimestri:

1. Creare con python un grafico che mostri l'andamento delle medie orarie per ciascun commesso; sul grafico vanno riportati simultaneamente **tutti i venditori**, con una legenda esplicativa.
2. Fare un grafico simile che riporti l'andamento del voto di ciascun commesso in funzione del tempo.
3. A partire dai due grafici precedenti, qual è il commesso che ha ottenuto la performance *media* migliore su ciascun trimestre? Quale la peggiore? Quale dei commessi ha dimostrato un margine di miglioramento e quale è peggiorato in maniera significativa? Scrivere le risposte sul file risposte.txt.

Dati Annuali di vendita (Primo, Secondo, Terzo, Quarto trimestre)

1	450	32	450	32	360	28	500	40
2	322	14	380	31	415	31	324	60
3	870	82	900	52	320	15	400	50
4	765	76	800	60	643	28	300	25
5	198	20	600	80	472	32	120	10
6	145	10	744	61	432	22	374	18
7	828	75	123	12	328	19	702	40
8	932	80	426	37	544	26	187	21

9 267 32 184 14 612 43 781 72
10 424 18 212 22 374 32 643 47

Suggerimento
Modificare la matrice dati in modo che abbia 8 colonne. Leggere tutti i dati dal file anno2016.dat e poi usando un ciclo for sui quattro trimestri, fare esattamente quello che si è fatto nella seconda parte (calcolare le medie del trimestre, usare la matrice sortmedia per fare il bubblesort per un dato trimestre, e dare un voto ai commessi per quel trimestre.) facendo attenzione ad usare le colonne giuste della matrice dati.