LABORATORIO FONDAMENTI DI INFORMATICA 16-17 DICEMBRE 2013

Promemoria funzioni di libreria riguardanti i file:

```
FILE * fopen(char filename[], char mode[]); //w = write r = read
int feof(FILE *fp); //ritorna 1 se cursore e' alla fine del file
int fclose(FILE *fp); //chiude il file
int fgetc(FILE *fp); //legge un char
int fputc(int c, FILE *fp); //scrive un char
int fprintf(FILE *fp, char format[], ...); //scrive testo
int fscanf(FILE *fp, char format[], ...); //legge testo
int fread(&record, sizeof(record), 1, fp); //lettura binaria
int fwrite(&record, sizeof(record), 1, fp); //scrittura binaria
long ftell(FILE *fp); //restituisce la posizione nel file
int fseek(FILE *fp, long offset, int whence); //imposta la posiz.
```

Esercizio 1

Scrivere un programma che continui a chiedere tramite la funzione `gets` delle stringhe `s` di dimensione massima `DIM=100` caratteri all'utente fino a che quest'ultimo non inserisce la parola "`fine`". Se l'utente inserisce la parola "`indietro`", significa che desidera tornare indietro e sovrascrivere l'ultima parola inserita. Tutte le parole (compresa la parola "`fine`", esclusa la parola "`indietro`") vanno mano a mano scritte in un file di testo tramite la funzione `fprintf`. Il nome del file di testo va definito tramite la costante `NOMEFILE` (`#define NOMEFILE "file.txt"`). Il file va aperto tramite la funzione `fopen` in modalità scrittura "`w`" e va chiuso tramite la funzione `fclose` prima di terminare il programma. Prima di effettuare una scrittura, occorre salvare la posizione attuale in byte nel file nella variabile "`long int pos;`" tramite la funzione `ftell`. Se l'utente inserisce la parola "`indietro`", occorre riportare indietro la posizione nel file utilizzando la funzione "`fseek(fp, pos, SEEK_SET);`".

Nota: utilizzare l'istruzione "`printf("%ld", pos);`" se si desidera stampare la variabile `pos`

Nota per i più curiosi: l'istruzione "`truncate(NOMEFILE, pos);`" permette di troncare il file alla posizione `pos` ("`truncate`" si trova in "`<unistd.h>`").

```
Apro il file "file.txt" ...
Quale parola vuoi scrivere nel file? Buone
Ok, scrivo 'Buone'. Cursore passato dal byte 0 al byte 6.
Quale parola vuoi scrivere nel file? Feeste
Ok, scrivo 'Feeste'. Cursore passato dal byte 6 al byte 13.
Quale parola vuoi scrivere nel file? indietro
Ritorno al byte 6.
Quale parola vuoi scrivere nel file? Feste
Ok, scrivo 'Feste'. Cursore passato dal byte 6 al byte 12
Quale parola vuoi scrivere nel file? fine
Ok, scrivo 'fine'. Cursore passato dal byte 12 al byte 17
Chiudo il file 'file.txt'...
```

CONTENUTO DEL FILE:

```
Buone
Feste
fine
```

Esercizio 2

La seguente funzione genera un testo casuale e lo inserisce nella stringa "s" passata come parametro. Il testo è composto da un numero casuale `len` di caratteri maiuscoli da `'A'` a `'Z'`, con `1 <= len <= (DIM - 1)`, ed è terminato dal carattere '\0'.

```
void genera(char s[DIM]) {
        int i, len;
        len = rand() % (DIM - 1) + 1;
        for (i = 0; i < len; i++) {
              s[i] = 'A' + rand() % ('Z' - 'A' + 1);
        }
        s[i] = '\0';
}
```

Tramite un programma, le stringhe generate dalla funzione "`genera`" vanno inserite una di seguito all'altra in un file di testo denominato `#define NOMEFILE "output.txt"`. Tale file deve però essere formattato in righe con lunghezza massima `MAX=20` caratteri; pertanto, se l'aggiunta di una nuova stringa dovesse far eccedere la lunghezza massima, tale stringa deve essere inserita su una nuova riga del file (cioè dopo il carattere di "a capo" '\n').

Ogni riga del file deve terminare con il numero di caratteri di cui è composta la riga e il numero della riga.

La composizione del testo termina quando il file "`output.txt`" contiene `N_RIGHE=10` righe di testo.

Utilizzare la funzione "`fprintf`" per scrivere le stringhe e i numeri sul file "`output.txt`".

```
Generazione delle stringhe e scrittura del file...
'RFKQY' 'QFJKXYQ' 'N'
'TYSFRZRM' 'LYGFVEU' 'QF' 'DB'
...
'TAKCZ' 'HSYB' 'OETSWCR'
```

CONTENUTO DEL FILE:

```
RFKQYQFJKXYQN num_caratteri=13 riga=1
TYSFRZRMLYGFVEUQFDB num_caratteri=19 riga=2
...
TAKCZHSYBOETSWCR num_caratteri=16 riga=10
```

Esercizio 3

Scrivere un programma che dato un file di testo, ne crei una copia in cui i caratteri minuscoli vengano convertiti in maiuscoli (`'a'` → `'A'`) e viceversa i caratteri maiuscoli vengano convertiti in minuscoli (`'A'` → `'a'`).

Creare preliminarmente tramite un editor di testo il file sorgente salvandolo con il nome "`sorgente.txt`" e inserendovi un contenuto a piacere, ad esempio "`Felice Anno Nuovo 2014`".

Creare e poi richiamare dalla funzione "`main`" una funzione con questo prototipo:

```
void copiafile (char * orig, char * copia);
```

Il parametro "`orig`" contiene il nome del file sorgente, mentre il parametro "`copia`" contiene il nome del file destinazione. E' possibile utilizzare la funzione "`fgetc`" per leggere un carattere dal file sorgente e la funzione "`fputc`" per scrivere un carattere nel file di copia. Usare la funzione "`feof`" per capire se il file sorgente è stato completamente letto. Ricordare la corrispondenza caratteri ↔ numeri interi e che la "distanza" tra un carattere maiuscolo e un carattere minuscolo è data dalla differenza `'a' - 'A'`.

```
sorgente.txt:  Felice Anno Nuovo 2014
copia.txt:     fELICE aNNO nUOVO 2014
```

Esercizio 4

La seguente funzione genera un numero intero `n` con `-10 <= n <= +10`

```
int genera_num() {
     return rand() % 21 - 10;
}
```

Scrivere un programma che una volta generata una matrice quadrata di numeri interi di dimensione `DIM` x `DIM` (con `DIM = 3`) tramite la funzione "`genera_num`", sia in grado di memorizzare e rileggere tale matrice in e da un file binario denominato `#define FILEBIN "matrice.dat"`.

Per essere salvati nel file binario gli elementi della matrice devono prima essere inseriti in una variabile di tipo strutturato che contenga il `valore` assoluto (sempre positivo) dell'elemento, il suo `segno` (carattere `'P'` per positivo, `'N'` per negativo) e gli indici di `riga` e `colonna` a cui si trova l'elemento.

```
typedef struct {
     int valore; //valore >= 0
     char segno; //'P' se il valore e' positivo, 'N' se negativo
     int riga, colonna; //indici di riga e colonna nella matrice
} s_elem;
```

Per realizzare il programma creare e lanciare dalla funzione "`main`" le seguenti tre funzioni (si consiglia la seguente sequenza: generazione della matrice, stampa, scrittura su file, lettura dal file e ristampa per verificare che il processo di scrittura/lettura sia andato a buon fine)

```
void stampamatrice(int m[DIM][DIM]); //stampa a video la matrice

void scrivimatrice(int m[DIM][DIM]); //scrive m sul file binario
```

Per scrivere la variabile "`record`" sul file binario utilizzare la funzione di libreria

```
     fwrite(&record, sizeof(record), 1, fp);
```

```
void leggimatrice(int m[DIM][DIM]); //legge dal file binario m
```

Per leggere la variabile "`record`" dal file binario utilizzare la funzione di libreria

```
     int num_letti = fread(&record, sizeof(record), 1, fp);
```

La funzione restituisce il numero di elementi che è riuscita a leggere pertanto quando restituisce 0 la lettura è terminata.

```
Matrice 'm':
 -3  -3  -8  10   3
 10  -1  -2  -2  -9
 -4  -8 -10  10 -10
 -5   6  -6  -6   0
  8   9   5   4  -9
Scrivo sul file 'matrice.dat' la matrice 'm'...
Leggo dal file 'matrice.dat' la matrice 'm2'...
Matrice 'm2':
 -3  -3  -8  10   3
 10  -1  -2  -2  -9
 -4  -8 -10  10 -10
 -5   6  -6  -6   0
  8   9   5   4  -9
```