# Esercitazioni di Fondamenti di Informatica - Lez. 8 12/11/2020

## Esercizi sulla gestione dei file in C

*I codici e le soluzioni degli esercizi sono nella cartella codice8*

### Funzioni di libreria usate

*feof(FILE *stream)* testa il flag end-of-file per lo stream

EOF è un carattere speciale (ASCII 26) che indica che è stato raggiunto il fine-file e può essere generato dalla tastiera con Ctrl-Z .
Quando viene creato un file, il carattere speciale EOF viene inserito dal sistema operativo dopo l'ultimo carattere del file.

*fwrite()* scrive su stream nmemb elementi,
ciascuno di dimensione size. Il primo elemento e' individuato dal puntatore ptr e
ritorna il numero di elementi scritti.

*char *fgets(char *s, int size, FILE *stream)*
legge una linea dallo stream immagazzinandola nel buffer puntato da s. Sono letti al piu' (size - 1) caratteri, oppure fino al raggiungimento del carattere di new-line '\n' o di EOF

*int fseek(FILE *stream, long offset, int whence)*
La funzione fseek() imposta l'indicatore di posizione del file. La prossima operazione di I/O su stream verra' eseguita dalla nuova posizione impostata.
La posizione e' calcolata aggiungendo offset (che puo' assumere anche valori negativi) a whence. whence puo' valere SEEK_SET, SEEK_CUR o SEEK_END per specificare rispettivamente il riferimento dall'inizio file, dalla posizione corrente o dalla fine file.
In caso di successo, fseek() ritorna 0 e viene cancellato l'indicatore di fine file. In caso di fallimento ritorna -1.

*ftell()*
ritorna il valore corrente dell'indicatore di posizione del file associato a stream.

## Introduzione

Creare una funzione C che legga tutto ciò che viene scritto sullo standard input e riporti ogni carattere letto su un  file passato come argomento.
La lettura continua fino alla lettura di un carattere terminatore '$'

un algoritmo può essere il seguente:

*c='/0'*

*file f1*

*finchè c diverso da '$'*

*leggi c*

*scrivi c in file f1*

end

implementazione:

```
#include<stdio.h>
#include<stdlib.h>

void leggi_stdin(FILE *file){

 char c;
 do{
       c=getchar();
       if(c!=EOF && c!='$'){
             fputc(c,file);
       }
 }while(c!='$');
}

int main(){

 FILE *file;

 if((file = fopen("log.txt","w"))==NULL){
       puts("Non e' stato possibile aprire il file");
       exit(1);
 }
 leggi_stdin(file);
 fclose(file);
 printf("Il file log.txt è stato creato \n");
}
```

1. **Scrivere una funzione C che prende come parametro un file e stampa a video tutte le sue righe dall'ultima alla prima. Ogni riga è lunga non più di 120 caratteri.**

```c
#include<stdio.h>
#include<stdlib.h>
#define MAX_READ 120

/* funzione ricorsiva */

void contrario(FILE * file){

    if(!feof(file)){

          char stringa[MAX_READ];
          fgets(stringa,MAX_READ,file);


    /* Chiamata ricorsiva
       N.B. ogni volta che effettuiamo la
          lettura il puntatore a file
          viene spostato avanti di MAX_READ caratteri
          nel nostro caso leggiamo una riga per volta
          dato che non abbiamo righe con più di 120 caratteri
    */
    contrario(file);

    /* al termine della chiamata ricorsiva stampiamo la
       riga letta
       N.B. siccome la printf e' dopo la chiamata ricorsiva
       inizieremo a stampare dalla fine del file
    */
    printf("%s",stringa);
    }}
```

2. **Data la struttura Rubrica, creare una funzione C che dati come parametri una rubrica e un file aperto in modalità binaria salvi sul file l'array di contatti e una funzione che dato un file e una rubrica, recuperi dal file i contatti salvati**

```c
#define MAX_CONTATTI 100
#define MAX_NOME 30
#define MAX_TEL 11

typedef struct{

 char nome[MAX_NOME];
 char telefono[MAX_TEL];
} Contatto;
```

```
typedef struct{

 Contatto contatti[MAX_CONTATTI];
 int numero_contatti;
} Rubrica;
```

soluzione

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

#define MAX_NOME 30
#define MAX_TEL 11
#define MAX_CONTATTI 100

typedef struct{
 char nome[MAX_NOME];
 char telefono[MAX_TEL];
} Contatto;

typedef struct{
 Contatto contatti[MAX_CONTATTI];
 int numero_contatti;
} Rubrica;

/* Salva l'array di contatti su un file binario*/

void salva(FILE *file, Rubrica rubrica){


fwrite(rubrica.contatti,sizeof(Contatto),rubrica.numero_contatti
,file);

}



/* Leggere l'array di contatti da un file binario */
void ripristina(FILE*file, Rubrica * rubrica){

 /* Recupero il numero di contatti in base alla
    dimensione del file
 */
 fseek(file,0,SEEK_END);


 rubrica->numero_contatti = ftell(file)/sizeof(Contatto);

 /*
 Ricomincio la lettura dall'inizio del file
 */
 rewind(file);

 /* Leggo i contatti */
 fread(rubrica->contatti,sizeof(Contatto),rubrica-
>numero_contatti,file);
}

int main(){
```

```
    Rubrica rubrica1 = {
                    {{"Mario", "000000"},
                     {"Filippo", "000001"},
                     {"Giulia", "000002"}},
                          3
                          };
    FILE *file;
    if((file=fopen("rubrica.bin","wb"))==NULL){
          puts("Non e' possibile aprire il file in scrittura");
          exit(1);
    }
    salva(file,rubrica1);
    fclose(file);

    /* Dichiaro un'altra struttura rubrica */
    Rubrica rubrica2;
    int i;
    FILE * file2;
    if((file2=fopen("rubrica.bin","rb"))==NULL){
                    puts("Non e' possibile aprire il file in
lettura");
                    exit(1);
           }
    ripristina(file2,&rubrica2);
    fclose(file2);
    for(i=0;i<rubrica2.numero_contatti;i++){
          printf("%s -
%s\n",rubrica2.contatti[i].nome,rubrica2.contatti[i].telefono);
    }
}
```

3. **Dato un file di testo aperto in lettura, controllare se il testo contenuto e' un palindromo.**

```
#include<stdio.h>
#include<stdlib.h>

typedef enum{FALSE,TRUE} bool;

bool palindromo(FILE * file){

 int count = 0;
 char inizio, fine;

  bool res = TRUE;
  rewind(file);

 /*
 Leggo un carattere all'inizio del file
 */
 while((inizio=fgetc(file) )!=EOF){
          count ++;

          //  Mi sposto alla fine del file
```

```
            fseek(file,-count*sizeof(char),SEEK_END);
            /* N.B. a questo punto count e' stato
               incrementato, quindi non leggero'
               EOF come primo carattere alla prima lettura*/

            /* Leggo un carattere */
            fine = fgetc(file);
            if(inizio!=fine){
                  return FALSE;
            }
            /*
            Mi riposiziono all'inizio del file
                  count e' già stato incrementato
            */
            fseek(file,count*sizeof(char),SEEK_SET);
            /*
            N.B.  Quando mi riposiziono all'inizio
                  o alla fine del file devo usare
                  un contatore per sapere quanti
                  caratteri saltare
            */
      }
      return TRUE;
     // return res;
     }

     int main(){

      FILE * f;

      if ((f=fopen("palindromo.txt","r"))==NULL){
            puts("Non e' possibile aprire il file");
            exit(1);
      }

      if(palindromo(f)){
            printf("Testo palindromo\n");
      }
      else{
            printf("Testo non palindromo\n");
            fclose(f);
      } }
```

4. **Scrivere una funzione C che ha come unico parametro un file testuale contenente una lista di interi (intervallati da uno spazio), dove il primo intero indica quanti interi sono contenuti. La funzione deve leggere gli interi e aggiungere alla media dei valori letti.**
   **Dare anche un esempio del main chiamante la funzione**

```c
#include<stdio.h>
#include<stdlib.h>

int media(FILE *file){

 int numero_interi, indice , n;
 double media = 0;

 rewind(file);

 /*
 Leggo il numero n di interi contenuti
 */
 fscanf(file,"%d",&numero_interi);

 /*
 Leggo iterativamente n interi
 */
 for(indice=0; indice<numero_interi; indice++){
       fscanf(file,"%d",&n);

       media+=n;
 }
 media /=numero_interi;
 /*
 Stampo la media in un formato appropriato
 */
 //fprintf(file," %.2f",media);
return media;
}


int main(){

 FILE *f;
 if ((f=fopen("media.txt","r+"))==NULL){
       puts("Errore apertura file");
       exit(1);
 /* r+ apre un file e abilita update
    w+ apre file e cancella i contenuti
       se non esiste lo crea
 */
 }
 media(f);

 fclose(f);
}
```

5. **Scrivere una funzione C che prende come argomenti una parola e un file (di testo) ed effettua la ricerca di tale parola all'interno del file. Il testo contiene solo lettere e spazi. La funzione deve restituire il numero di occorrenze della parola.**

   **Es: la ricerca della parola 'asso' nel seguente testo**

   **'ASSO asso sasso asso'**

   **deve restituire 2 come valore di ritorno della funzione.**

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

/*Funzione che recupera una parola dal file
(alternativa alla fscanf di una stringa)

void sottostringa(FILE *file, char *parola) {
  int indice = 0;
  char c;
  do {
    c = fgetc(file);
    if(c == ' ' || c == EOF)
      parola[indice] = '\0';
    else
      parola[indice] = c;
    indice ++;
  } while(c != ' ' && c != EOF);
}
*/

int ricerca(char * parola,FILE*file){
   char s[100];
   int n=0;

   rewind(file);
   do{
        //sottostringa(file,s);
         // invece di fscanf

         fscanf(file,"%s",s);

         /*
         Confronto la parola da cercare
         e quella letta
         */
         if(strcmp(s,parola)==0){
               n++;
         }
         printf("%d %s\n",strcmp(s,parola),s);
   }while(!feof(file));
```

```
    return n;
}

int main(){

    FILE * file;

    if((file = fopen("ricerca.txt","r"))==NULL){
          puts("Errore apertura file");
          exit(1);
    }
    printf("Numero occorrenze: %d\n",ricerca("asso",file));

    fclose(file);
}
```

6. **Scrivere in C una funzione che legge un file di numeri interi e restituisce un nuovo file (numeriPari). Questo deve contenere i soli numeri pari letti, ma memorizzati in ordine inverso.Ad esempio, se il file originario contenesse: 1, 2, 3, 4, 5, 6, 7, 8, il nuovo file dovrebbe contenere: 8, 6, 4, 2. Il parametro della funzione e il valore restituito non devono essere nomi di file, mai i file stessi.**

```
#include<stdio.h>
#include<stdlib.h>

/* Funzione ricorsiva ausiliaria */
void numPari_ricorsiva(FILE *input, FILE *output){
 int i;
 /* Legge un 'intero*/
 if(fread(&i,sizeof(int),1,input)!=0){
        /* Richiama se stessa in modo ricorsivo (stampa quindi prima
il prossimo intero)*/
        numPari_ricorsiva(input,output);
        if(i%2==0){
        /* Stampa l'intero pari letto dopo aver terminato la chiamata
ricorsiva */
        fwrite(&i,sizeof(int),1,output);
        }
 }
}

/* Funzione principale che crea il file di output e richiama la
funzione ricorsiva */
FILE *numPari(FILE *input){
 FILE *output;

 if((output = fopen("numeriPari","rb+")) == NULL){
        return NULL;
 }
 rewind(input);
 numPari_ricorsiva(input,output);
 return output;
```

```
}



int main(){

 FILE *input, *output;
 int n;

        /* Generiamo un file binario contenente gli interi
           da 1 a 100
        */
        if((output=fopen("numeri_contrario","wb"))==NULL){
                        puts("Errore lettura file");
                        exit(1);
         }
         int i;
         for(i=1;i<=100;i++){
           fwrite(&i,sizeof(int),1,output);
         }
         fclose(output);

        /*
          Stampiamo a stdin il file appena generato
        */

        if((input=fopen("numeri_contrario","rb"))==NULL){
              puts("Errrore lettura file");
              exit(1);
        }

        /* Stampiamo il file di input */
        printf("INPUT\n");
        while(fread(&n,sizeof(int),1,input)!=0){
                                printf(" %d ",n);
                        }

        output = numPari(input);
        fclose(input);


        /* stampiamo il file generato */
        if(output!=NULL){
              printf("\n OUTPUT \n");
              rewind(output);
              fread(&n,sizeof(int),1,output);
              while
              (fread(&n,sizeof(int),1,output)!=0){
                    printf(" %d ",n);
              }
              printf("\n");
              fclose(output);
        }

}
```

**7. Scrivere una funzione C che dato un file di interi e un intero n (entrambi passati come parametri), modifichi i numeri contenuti nel file dividendoli per n. Il risultato di ogni divisione deve essere arrotondato al primo intero inferiore**

```c
#include<stdio.h>
#include<stdlib.h>
#define NUM_LEN 10

void dividi(FILE *file, int n){

 int i;
 /* Legge tutti i numeri
    contenuti nel file
 */
 while(fread(&i,sizeof(int),1,file)!=0){
        i/=n; /* Divide il numero letto*/
        /*
        Ritorna indietro nel file
        */
        fseek(file,-sizeof(int),SEEK_CUR);
        /*
        E sovrascrive il numero letto
        */
        fwrite(&i,sizeof(int),1,file);
 }
}

int main(){

 FILE *file;
        /*
        Generiamo un file binario con gli interi da 1 a 10
        */

        int numbers[NUM_LEN] ={1,2,3,4,5,6,7,8,9,10};

        if((file = fopen("dividi","wb+"))==NULL){
              puts("Errore lettura file");
              exit(1);
        }
        fwrite(numbers,sizeof(int),NUM_LEN,file);

        /* Stampiamo il contenuto del file */
        rewind(file);
        int a[NUM_LEN];
        fread(a,sizeof(int),NUM_LEN,file);
        for(int i=0;i<NUM_LEN;i++){
              printf(" %d ",a[i]);
        }
        fclose(file);

        /* apriamo il file appena generato*/
        if((file = fopen("dividi","rb+"))==NULL){
              puts("Errore lettura file");
```

```
                exit(1);
            }
        /*
        eseguiamo la funzione dividi sul file  e stampiamo il risultato
        */
        dividi(file,3);
        rewind(file);
          int i;
          printf("\n");
          while(fread(&i,sizeof(int),1,file)!=0){
              printf(" %d ",i);
         }

        fclose(file);
        printf("\n");
    }
```