Esercizi sui file

Ricerca di stringhe in un file di testo.

Dato un file F, si realizzi un programma che, acquisita una parola P da standard input, ricerchi P all'interno del file dato e stampi sul dispositivo standard di output il numero delle occorrenze di P nel file F.

```
#include <stdio.h>
#include <string.h>
#define MAX 81
/* ricerca di parole in un file di testo */

main()
{ FILE *f1;
  char P[MAX], S[MAX];
  int cont=0;
  f1=fopen("D:\\exC\\F1.txt", "r");
  if (f1==NULL){
       printf("Errore apertura del file!\n");
       return;
  }
  printf("\nParola da cercare ? ");
  scanf("%s", P);
  fscanf(f1, "%s", S);
  while (!feof(f1))
  {
   if (strcmp(S,P)==0){
        printf("\nHo trovato %s!\n", P);
       cont++;
   }
   fscanf(f1, "%s", S);
  }
  printf("Trovata %s %d volte \n", P, cont);
fclose(f1);
}
```


Il nome del file viene preso come percorso assoluto. Si noti il doppio simbolo '\' (serve per far funzionare il programma sotto Rhide). Il simbolo '\' infatti è un simbolo speciale del C che per essere interpretato come carattere '\' deve essere raddoppiato.

Il file, essendo un file testo, può essere creato con un qualunque text editor.

L'apertura può fallire (per esempio nel caso in cui il file non esista).

Viene poi chiesta all'utente la parola da cercare che viene memorizzata nella variabile `P`.

Successivamente, si procede alla lettura del file, parola per parola, utilizzando una sequenza di `fscanf`.

In particolare, la prima `fscanf` legge la prima parola contenuta nel file.

Ogni iterazione del successivo `while` effettua la lettura di una nuova parola; la ripetizione è controllata dal risultato della chiamata alla funzione standard `feof(f1)`:

- se il risultato è *vero* (valore restituito >0), significa che l'ultima lettura non è stata eseguita perché è stata incontrata la fine del file
- se, invece, `feof(f1)` restituisce falso (cioè il valore 0), significa che l'ultima lettura è stata eseguita con successo.

Ogni parola letta viene confrontata, utilizzando la funzione `strcmp`, della libreria `<string.h>` con la parola `P`: se `P` ed `S` sono uguali, la `strcmp` restituisce il valore 0, e viene stampato un messaggio ed incrementato il contatore `cont`; poi si passa a leggere la parola successiva dal file.

Al termine della ripetizione viene stampato il numero delle occorrenze di `P` (rappresentato dal valore di `cont`) ed infine il file viene chiuso (con la funzione `fclose`).

Creazione di un file binario

Si realizzi un programma che costruisca un file binario di nome "anagrafe" contenente le informazioni relative ai cittadini di un comune italiano.
Ogni record del file "anagrafe" rappresenta una persona abitante nel comune e contiene le seguenti informazioni:

- Nome: una stringa che rappresenta il nome della persona;
- Cognome: una stringa che rappresenta il cognome della persona;
- Data di nascita: un insieme di tre interi che rappresenta la data di nascita;
- Luogo di nascita: una stringa che rappresenta la località di nascita;
- Indirizzo: una stringa che rappresenta l'indirizzo della persona;
- Codice Fiscale: una sequenza di 16 caratteri che individua univocamente il cittadino.

Il programma dovrà riempire il file "anagrafe" con informazioni immesse dall'utente attraverso lo standard input e, successivamente, visualizzarne il contenuto.

Per risolvere il problema è necessario predisporre un tipo di dato adatto a rappresentare il record logico del file da creare:

```
struct elemento
{       char nome[50];
        char cognome[50];
         char l_nascita[50];
        char indirizzo[80];
        char CF[16];
};
```


Il programma è suddiviso in due funzioni:

```
void creafile(char *v);
void vedifile(char *v);
```

che servono rispettivamente a creare e a visualizzare il contenuto del file binario il cui nome **`v`** è dato come parametro.

La struttura del main è quindi la seguente:

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

struct elemento {
   char nome[50];
   char cognome[50];
   char l_nascita[50];
   char indirizzo[80];
   char CF[16];
};

/* prototipi procedure usate nel main */
void creafile(char *v);
void vedifile(char *v);


main()
{   creafile("D:\\exC\\anagrafe.dat");
    vedifile("D:\\exC\\anagrafe.dat");
}
```

La funzione `creafile` viene definita come segue:

```
/*prima di creafile dichiaro il prototipo
  della funzione leggiel() che non ha
  parametri e restituisce un valore di tipo
  struct elemento. */
struct elemento leggiel();

void creafile(char *v) {
FILE *f; struct elemento e; int fine=0;

  f=fopen(v, "wb");
  printf("Creazione di %s \n", v);
  while (!fine)
    { e=leggiel();
      fwrite(&e,sizeof(struct elemento),1,f);
      printf("\nFine (SI=1, NO=0) ? ");
      scanf("%d", &fine);
    }
  fclose(f);
}
```

creafile esegue le seguenti operazioni:

- apre in scrittura il file binario di nome **v**
- esegue un ciclo **while**:
  - ad ogni iterazione la funzione acquisice i dati relativi ad un nuovo elemento **e** mediante la funzione **leggiel**;
  - l'elemento letto viene poi inserito nel file con la funzione standard **fwrite**.
  - al termine di ogni iterazione, l'utente specifica (mediante un intero assegnato alla variabile fine) se terminare la scrittura o inserire un nuovo elemento.

L'ultima operazione da compiere è la chiusura del file, effettuata con **fclose**.

Vediamo ora la definizione della funzione **leggiel**:

```
struct elemento leggiel() {
struct  elemento e;
        printf("Nome ? ");
        scanf("%s", e.nome);
        printf("\nCognome ? ");
        scanf("%s",e.cognome);
        printf("\nLuogo di nascita ? ");
        scanf("%s",e.l_nascita);
        printf("\nIndirizzo ? ");
        scanf("%s",e.indirizzo);
        printf("\nCodice Fiscale ? ");
        scanf("%s", e.CF);
        return e;
}
```

La funzione **leggiel** legge dallo standard input i dati relativi ad un nuovo elemento **e** ed assegna i valori letti ai rispettivi campi della variabile locale **e**.

Vediamo ora la definizione della funzione **vedifile**:

```
/*prima di vedifile dichiaro il prototipo
  della funzione stampael() che ha come
  parametro un elemento di tipo struct
  elemento e restituisce un valore di tipo
  struct elemento. */

void stampael(struct elemento e);

void vedifile(char *v)
{       FILE *f; struct elemento e;
 f=fopen(v, "rb");
 printf("Lettura di %s:\n", v);
 fread(&e, sizeof(struct elemento), 1, f);
 while (!feof(f))
  { stampael(e);
   fread(&e, sizeof(struct elemento),1,f);
  }
  fclose(f);
}
```

La funzione inizialmente apre il file data come parametro il lettura (modalità **"rb"**).

Mediante la successiva istruzione **while**, ad ogni iterazione viene estratto un elemento alla volta dal file utilizzando la funzione standard **fread** (perchè il file è binario).

Ogni elemento letto **e** viene poi stampato mediante l'uso della funzione **stampael** definita come segue:

```
void stampael(struct elemento e)
{
        printf("%s\t", e.nome);
        printf("%s\t", e.cognome);
        printf("%s\t",e.l_nascita);
        printf("%s\t", e.indirizzo);
        printf("%s\n", e.CF);
}
```

Ricerca di record in un file binario

Realizzare un programma che, dato il file binario "anagrafe" generato con il programma presentato precedentemente, ricerchi e stampi tutti gli elementi del file che hanno cognome uguale ad una stringa C letta da standard input. Il programma deve inoltre stampare il numero totale degli elementi del file che hanno il cognome uguale a C.

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

struct elemento{
        char nome[50];
        char cognome[50];
        char l_nascita[50];
        char indirizzo[80];
        char CF[16];
};

void stampael(struct elemento e);

main()
{ FILE *f;
 struct elemento e;
 char C[50];
 int k=0;
 printf("Inserire il cognome da ricercare: ");
 scanf("%s",C);
 f=fopen("D://exC//anagrafe.dat", "rb");
 while(fread(&e, sizeof(struct elemento), 1, f)>0)
     if (!strcmp(e.cognome, C))
     {  stampael(e);
        ++;
     }
 printf("\n %d cittadini con cognome %s\n",k, C);
 fclose(f); }
```



```
void stampael(struct elemento e) {
        printf("%s\t", e.nome);
        printf("%s\t", e.cognome);
        printf("%s\t",e.l_nascita);
        printf("%s\t", e.indirizzo);
        printf("%s\n", e.CF);
}
```

La struttura di ogni elemento è descritta dalla dichiarazione del tipo di dato **`struct elemento`**.

Il programma, dopo aver acquisito dallo standard input il cognome C da ricercare, apre il file binario "anagrafe" in lettura mediante la funzione standard **`fopen`**.

Successivamente, il ciclo **`while`** consente di leggere sequenzialmente il contenuto del file, elemento per elemento. La lettura, poiché il file è binario, viene effettuata mediante la funzione **`fread`**.

Il ciclo **`while`** è controllato dal valore ottenuto come risultato della chiamata alla funzione **`fread`**:

- se è stata incontrata la fine del file (e quindi la lettura non ha avuto successo) **`fread`** ritorna 0;
- se, invece, la lettura è stata effettuata correttamente, la funzione **`fread`** restituisce un valore positivo che rappresenta il numero di byte effettivamente letti.

Ad ogni iterazione il programma dispone quindi di un nuovo elemento **`e`** estratto dal file;

l'**`if`** interno al ciclo **`while`** controlla se il valore del campo **`cognome`** di **`e`** coincide con il cognome C immesso dall'utente. In caso affermativo, **`e`** viene stampato (mediante la funzione **`stampael`**) ed il contatore delle occorrenze **`k`** viene incrementato.

Infine, il programma stampa il numero delle occorrenze **`k`**, chiude il file e poi termina.

Ricerca degli elementi comuni a due file

Siano dati due file contenenti testo. Si realizzi un programma che individui e scriva un terzo file "risultati.txt" le parole che compaiono in entrambi i file.

```
#include <stdio.h>
#include <string.h>
#define MAX 81
/* intersezione di due file di testo */

main()
{
  FILE *f1, *f2, *f3;
  char P[MAX], S[MAX];
  int trovato=0;
  f1=fopen("D:\\exC\\f1.txt", "r");
  f2=fopen("D:\\exC\\f2.txt", "r");
  f3=fopen("risultati.txt", "w");
  while (fscanf(f1, "%s", S)>0)
   {rewind(f2);
    trovato=0;
    while (fscanf(f2,"%s",P)>0 && !trovato)
      if (strcmp(S,P)==0)
      {printf("\nHo trovato \"%s\"!\n", P);
       fprintf(f3, "%s\n", P);
       trovato=1;
      }
    }
   fclose(f1);
   fclose(f2);
   fclose(f3);
}
```