File di testo e file binari Soluzione 1 (2)

…continua

```
    printf("Cifrazione del file: ");
    scanf("%s",nome);
    // Controllo che il file esista.
    while((fs=fopen(nome,"r"))==NULL) {
        printf("Errore nel nome del file, file inesistente.\n");
        printf("Inserire il nome del file: ");
        scanf("%s",nome);
    }

    printf("Inserire una chiave (max %d char): ", DIM_K);
    scanf("%s", key);

    // Ora cifiramo il file, attenzione deve essere aperto!
    cryptFile(fs, "cifrato.txt", key);
    fclose(fs);

    printf("Operazione di cifrazione conclusa.\n\n");

    printf("Decifrazione del file: %s\n", "cifrato.txt");
    if((fs=fopen("cifrato.txt", "r"))==NULL) {
        printf("Errore nel nome del file, file inesistente.\n");
        exit(1);
    }

    decryptFile(fs, "decifrato.txt", key);
    fclose(fs);
    printf("Operazione di decifrazione conclusa.\n\n");
}
```

File di testo e file binari Soluzione 1 (3)

```
/*****************************************************************************
Critto il file, che è stato aperto nel puntatore fp, e lo inserisco nel file di
dest
inazione con nome pari a "nome" usando la chiave "key".
Al termine dell'operazione il file non è chiuso e l'operazione deve essere fatta
dal
main.
*****************************************************************************/
void cryptFile(FILE* fp, char* nome, char* key) {
    FILE* fd = fopen(nome, "w");
    int len = strlen(key);
    char c;
    int i;

    i=0;
    while(fscanf(fp, "%c", &c) != EOF) {
        c = c + key[i];
        fprintf(fd, "%c", c);
        i++;
        if(i >= len) i=0;
    }
    fclose(fd);
}
```

File di testo e file binari Soluzione 1 (4)

```
/*****************************************************************************
Decritto il file, che è stato aperto nel puntatore fp, e lo inserisco nel file di
dest
inazione con nome pari a "nome" usando la chiave "key".
Al termine dell'operazione il file non è chiuso e l'operazione deve essere fatta
dal
main.
*****************************************************************************/
void decryptFile(FILE* fp, char* nome, char* key) {
    FILE* fd = fopen(nome, "w");
    int len = strlen(key);
    char c;
    int i;

    i=0;
    while(fscanf(fp, "%c", &c) != EOF) {
        c = c - key[i];
        fprintf(fd, "%c", c);
        i++;
        if(i >= len) i=0;
    }
    fclose(fd);
}
```

File di testo e file binari Soluzione 2 (1)

Conversione da TXT a CSV

Un file di testo contiene i dati relativi a film (titolo, regista e anno), separati da spazi, e disposti su più righe (max 50). Scrivere un programma in C che legga il file di testo e importi i dati in un opportuno array di strutture.Si riscriva poi un secondo file (testo, con estensione .csv), in cui gli spazi tra i campi sono sostituiti da “,”.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXFILE 50

typedef struct {
     char titolo [20];
     char regia [20];
     int anno; } film ;

int leggi_file (film lista []){
     FILE * f;
     int i =0;
     f= fopen ("FILM.TXT ","rt");
     while (!feof(f)){ fscanf (f,"%s %s %d", lista[i].titolo ,lista[i].regia , &lista[i].anno);
          i++;
     }
     fclose (f);
     return i-1;
}
```

…continua

File di testo e file binari Soluzione 2 (2)

…continua

```
void scrivi_file (film lista[], int  nfilm){
    FILE * f;
    int i =0;
    f= fopen ("FILM.CSV ","wt");
    for (i=0;i<nfilm; i++)
    fprintf (f,"%s;%s;%d\n",lista[i].titolo ,lista[i].regia ,lista[i].anno);

    fclose (f);
}

main (){
    int nfilm;
    film listafilm [ MAXFILE ];
    nfilm = leggi_file (listafilm );
    scrivi_file (listafilm, nfilm);
}
```

File di testo e file binari Soluzione 1 (1)

Crypt-decrypt su file

Scrivere un programma in C che richiede all'utente il nome di un file di testo; apre il file in lettura e scrive sul file *crypt.txt* il cifrato (vedi lezione 6). Quindi apre il file *crypt.txt* e lo decifra scrivendo il risultato nel file *decrypt.txt*.
Confrontare il file di ingresso con il file *decrypt.txt.*

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

#define DIM_N  101
#define DIM_K  17


void cryptFile(FILE*, char*, char*);
void decryptFile(FILE*, char*, char*);

void main(void) {
    FILE* fs;
    char nome[DIM_N];
    char key[DIM_K];
```

…continua

File di testo e file binari Soluzione 3 (1)

Rubrica su file (`rubrica.rub`)

Ogni contatto è scritto secondo il seguente schema (fare uso delle typedef struct):

- nome (stringa di massimo 30 caratteri)
- cognome (stringa di massimo 30 caratteri)
- telefono (stringa di massimo 20 caratteri)
- età (intero)

Si scriva un programma in linguaggio C che legge il file e

- visualizza il contenuto del file
- legge da tastiera il nome di una persona
- chiede all'utente il nuovo numero di telefono
- aggiorna il dato corrispondente

visualizza di nuovo il contenuto del file

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

#define DIM_NOME   30
#define DIM_COGNOME 30
#define DIM_TEL      20
#define MAX     20

typedef struct {
        char cognome[DIM_COGNOME];
        char nome[DIM_NOME];
        char tel[DIM_TEL];
        int eta;
} elemento;
```

……continua

File di testo e file binari Soluzione 3 (2)

…continua

```
void stampa(FILE* f) {
        elemento e;
    printf("Stampo rubrica...\n");
    while(fread(&e, sizeof(elemento), 1, f) > 0) {
        printf("%s\t%s\n%s\n%d\n\n", e.cognome, e.nome,  e.tel, e.eta);
    }

}

void main(){
    elemento rub[MAX];
    char scelta;
    char nome[DIM_NOME];
    elemento e;
    int i=0, tot=0;

    FILE* f; /* dichiaro il file che andrò ad usare */

    f = fopen("rubrica.rub", "rb");
    if (f==NULL) {
        printf("Errore file lettura!");
        exit(1);
    }
    stampa(f);
    fclose(f);
```

…continua

File di testo e file binari Soluzione 3 (3)

```
…continua
     f = fopen("rubrica.rub", "rb");
     while(fread(&e, sizeof(elemento), 1, f) > 0) {
          rub[tot] = e;
          tot++;
     }
     fclose(f);
     //ricerca
     printf("Inserisci il nome da ricercare: ");
     scanf("\n%s", nome);
     while((i<tot) && (strcmp(rub[i].nome, nome))) i++;
     printf("Inserisci il nuovo telefono: ");
     scanf("\n%s", rub[i].tel);
     //riscrivo tutto l'array sul file
     f = fopen("rubrica.rub", "wb");
     if (f==NULL) {
          printf("Errore file scrittura!");
          exit(1);
     }
     fwrite(rub, sizeof(elemento), tot, f);
     fclose(f);
     f = fopen("rubrica.rub", "rb");
     if (f==NULL) {
          printf("Errore file lettura!");
          exit(1);
     }
     stampa(f);
     fclose(f);
}
```

File di testo e file binari Soluzione 4 (1)

Monitoraggio temperatura

Questo esercizio è costituito da due parti principali: il monitor che genera dei dati relativi ad una temperatura, e l'allarme che cataloga i dati rilevando le situazioni di allerta secondo una certa soglia. Le due parti vengono eseguite sequenzialmente: prima il monitor e poi l'allarme.
In particolare: Il monitor genera un file binario (uso di struct) che contiene dati di rilevamento temperatura (ora, temperatura). I dati sono generati in maniera casuale usando la funzione `rand()`. Vengono generati un totale di 24 dati (si suppone rilevamento quotidiano). Una volta che il monitor ha generato tutti i dati, entra in funzione l'allarme. Quest'ultimo richiede all'utente una soglia di sicurezza (una temperatura) oltre la quale i rilevamenti vengono considerati come situazione di allarme. Quindi apre il file generato dal monitor, legge tutti i valori copiando su un file di testo tutti i dati (ora e temperatura) la cui temperatura supera la soglia. In seguito viene stampato il file.

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

#define DATI        24
#define FILEMONITOR "rilevamenti.dat"
#define FILEALLARME "allarme.txt"

typedef struct r {
     int ore;
     float temp;
} rilevamento;

void monitor();
void stampa_monitor();
void allarme();
void stampa_allarme();
```

File di testo e file binari Soluzione 4 (2)

```
void main(){
     monitor();
     allarme();
     stampa_monitor();
     stampa_allarme();
}

void monitor(){

     rilevamento dato;

     int campione;
     FILE *fril;

     printf("Rilevamento dati in corso.\n");
     fril = fopen(FILEMONITOR, "wb");
     for(campione = 0; campione<DATI; campione++) {
          dato.ore = campione;
          dato.temp = (float)(rand()/650.0);
          printf(".");
          if(fwrite(&dato, sizeof(rilevamento), 1, fril) < 0) {
               printf("Errore!\n");
          }
     }
     fclose(fril);
}
```

File di testo e file binari Soluzione 4 (3)

```
void allarme() {

    rilevamento dato;

    int campione;
    float soglia;
    FILE *fril;
    FILE *fall;

    printf("\n\nCatalogazione dati.");
    printf ("\nInserisci la soglia si sicurezza: ");
    scanf("\n%f", &soglia);

    fril = fopen(FILEMONITOR, "rb");
    fall = fopen(FILEALLARME, "w");

    while((campione = fread(&dato, sizeof(rilevamento), 1, fril)) != 0) {
        if (dato.temp > soglia)
          fprintf(fall, "Ora:\t%d\tTemperatura:\t%2.1f C\n", dato.ore, dato.temp);
    }
    fclose(fall);
    fclose(fril);
}
```

File di testo e file binari Soluzione 4 (4)

```
void stampa_monitor() {

     rilevamento dato;

     int campione;
     FILE *fril;

     fril = fopen(FILEMONITOR, "r");
     printf("\n\nFile monitor. Rilevamenti registrati.\n");
     while((campione = fread(&dato, sizeof(rilevamento), 1, fril)) > 0) {
          printf("\nOra:\t%d\tTemperatura:\t%2.1f\tC", dato.ore, dato.temp);
     }
     fclose(fril);
}


void stampa_allarme() {

     FILE *fall;
     char rilevamento[50];

     printf("\n\nFile allarme. Situazioni di allarme.\n");

     fall = fopen(FILEALLARME, "r");
     while(fgets(rilevamento, 50, fall)!= NULL && !feof(fall)) {
          printf("%s", rilevamento);
     }
     fclose(fall);
}
```