ESEMPIO COMPLETO FILE BINARIO

È dato un file di binario **`people.dat`** i cui record rappresentano *ciascuno i dati di una persona*, secondo il seguente formato:

- **cognome** (al più 30 caratteri)
- **nome** (al più 30 caratteri)
- **sesso** (un singolo carattere, **`'M'`** o **`'F'`**)
- **anno di nascita**

Si noti che la creazione del file binario deve essere fatta da programma, mentre per i file di testo può essere fatta con un text editor.

CREAZIONE FILE BINARIO

Per creare un file binario e' necessario scrivere un programma che lo crei strutturandolo modo che ogni record contenga una **`struct`** di tipo **`persona`**

```
typedef struct
  {
  char cognome[31];
  nome[31];
  char sesso;
  int anno;
  } persona ;
```

I dati di ogni persona da inserire nel file vengono richiesti all'utente tramite la funzione **`leggiel()`** che non ha parametri e restituisce come valore di ritorno la **`struct`** di tipo **`persona`** letta. Quindi il prototipo e':

```
persona leggiel();
```

CREAZIONE FILE BINARIO

Mentre la definizione e':

```
persona leggiel()
{
 persona e;

 printf("Cognome ? ");
 fflush (stdin);
 scanf("%s", e.cognome);
 printf("\n Nome ? ");
 fflush (stdin);
 scanf("%s",e.nome);
 printf("\nSesso ? ");
 fflush (stdin);
 scanf("%c",e.sesso);
 printf("\nAnno nascita ? ");
 scanf("%d", e.anno);

 return e;
 }
```

CREAZIONE FILE BINARIO

Mentre la definizione e':

```
persona leggiel()
{
 persona e;

 printf("Cognome ? ");
 gets(e.cognome);
 printf("\n Nome ? ");
 gets (e.nome);
 printf("\nSesso ? ");
 scanf("%c",e.sesso);
 printf("\nAnno nascita ? ");
 scanf("%d", e.anno);

 return e;
 }
```

CREAZIONE FILE BINARIO

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
  {
  char cognome[31];
  nome[31];
  char sesso;
  int anno;
  } persona ;
persona leggiel()
int main()
{
FILE *f; persona e; int fine=0;
f=fopen("people.dat", "wb");
while (!fine)
     {
     e=leggiel();
     fwrite(&e,sizeof(persona),1,f);
     printf("\nFine (SI=1, NO=0) ? ");
     scanf("%d", &fine);
     }
  fclose(f);
}
```

CREAZIONE FILE BINARIO

L'esecuzione del programma precedente crea il file binario contenente i dati immessi dall'utente. Solo a questo punto il file può essere utilizzato.

Il file **`people.dat`** non e' visualizzabile tramite un text editor: questo e' il risultato

```
rossi >˜  ÿÿ @ˆ ˜ T ´  —8˜ ° ¨ ¯ â3
mario ˜ ˜ ôÜ¯ _˜ ˜ ôÜ¯ Aw O˜ Fˆ _˜ ¯
DÝ¯ M ¯ ¨ ¯ ¨ ¯ nuinH2ô1¨ ô1˜ ô1
```

ESEMPIO COMPLETO FILE BINARIO

Ora si vuole scrivere un programma che

- legga record per record i dati dal file
- e ponga i dati in un array di *persone*
- *(poi svolgeremo elaborazioni su essi)*

ESEMPIO COMPLETO FILE BINARIO

Come organizzarsi?

1) Definire una struttura di tipo **`persona`**

Poi, nel main:

2) Definire un array di strutture di tipo **`persona`**
3) Aprire il file in lettura
4) Leggere un record per volta, e porre i dati di quella persona in una cella dell'array

→ Servirà un indice per indicare la prossima cella libera nell'array.

ESEMPIO COMPLETO FILE BINARIO

1) Definire una struttura di tipo `persona`

Occorre definire una `struct` adatta a ospitare i dati elencati:

- **cognome** → array di 30+1 caratteri
- **nome** → array di 30+1 caratteri
- **sesso** → 1 carattere
- **anno di nascita** → un intero

ricordarsi lo spazio per il terminatore

```
typedef struct
  {
  char cognome[31];
  nome[31];
  char sesso;
  int anno;
  } persona ;
```

ESEMPIO COMPLETO FILE BINARIO

Poi, nel main:

2) definire un array di tipo **`persona`**
3) aprire il file in lettura

```
int main()
{
persona v[DIM];
FILE* f = fopen("people.dat", "r");
if (f!=NULL)
   {
   /* controllo che il file sia
      effettivamente aperto */
   }
  ...
 }
```

Hp: massimo DIM persone

ESEMPIO COMPLETO FILE BINARIO

Poi, nel main:

2) definire un array di tipo **persona**
3) aprire il file in lettura

```
int main()
{
persona v[DIM];
FILE* f = fopen("people.dat", "r");
if (f!=NULL)
   {
   /* controllo che il file sia
      effettivamente aperto */
   }
else
   {
   /* terminazione del programma */
   printf("Il file non esiste");
   }
}
```

Hp: massimo DIM persone

ESEMPIO COMPLETO FILE BINARIO

Poi, nel main:

4) leggere i record dal file, e porre i dati di ogni persona in una cella dell'array

Come organizzare la lettura?

```
int fread(addr, int dim, int n, FILE *f);
```

- legge dal file **n** elementi, ognuno grande **dim** byte (complessivamente, legge quindi n×dim byte)
- gli elementi da leggere vengono scritti in memoria a partire dall'indirizzo ***addr***

Uso fread

ESEMPIO COMPLETO FILE BINARIO

Poi, nel main:

4) leggere i record dal file, e porre i dati di ogni persona in una cella dell'array

Cosa far leggere a `fread`?

- *L'intero vettore di strutture: unica lettura per* **`DIM`** *record*

```
fread(v,sizeof(persona),DIM,f)
```

- *Un record alla volta all'interno di un ciclo*

```
i=0
while(!feof(f))
  {
  fread(&v[i],sizeof(persona),1,f);
  i++
  }
```

ESEMPIO COMPLETO FILE BINARIO

Poi, nel main:

4) leggere i record dal file, e porre i dati di ogni persona in una cella dell'array

Dove mettere quello che si legge?

- Abbiamo definito un array di tipo **`persona`** , **`v`**
- L'indice **`k`** indica la prima cella libera → **`v[k]`**
- Tale cella è una struttura fatta di *cognome*, *nome*, *sesso*, *anno* → ciò che si estrae da un record va direttamente nella struttura **`v[k]`**

ESEMPIO COMPLETO FILE BINARIO

```
#define DIM 30
#include <stdio.h>
#include <stdlib.h>
typedef struct
  {
  char cognome[31];
  nome[31];
  char sesso;
  int anno;
  } persona ;

int main()
{
persona v[DIM];
int i  ;
FILE* f;

FILE* f = fopen("people.dat", "r");
if (f!=NULL)
   {
   i=0
   while(!feof(f))
      {
      fread(&v[i],sizeof(persona),1,f);
      i++
      }
   }
else
   {
   printf("Il file non esiste!");
   }
}
```