13.5 Procedura anagrafica

Vediamo, a titolo di esempio, la gestione di un'anagrafica che abbiamo già esaminato nel Paragrafo 12.7 parlando di strutture dati composte tramite struttura, questa volta appoggiandoci su file in modo da rendere i dati persistenti (si veda il Listato 13.7), invece che lavorando in memoria tramite array di strutture.
Si osservi il modo in cui è gestito il file `anag.dat`. La funzione `ins_per()` apre il file:

```
fp = fopen("anag.dat","a+");
```

eventualmente creandolo, se non esiste, grazie all'opzione `a+`. Successivamente richiede all'utente i dati da inserire – cognome, nome, indirizzo ed età –, li memorizza nella struttura `anag` e provvede ad "appenderli" al file:

```
fwrite(&anag, sizeof(struct per), 1, fp);
```

La funzione di cancellazione `eli_per()`, invocata da `can_per()`, provvede a scrivere una struttura `anag` "vuota" su file, in corrispondenza della persona che deve essere eliminata. In questo caso l'apertura del file avviene con l'opzione `r+`:

```
fp = fopen("anag.dat","r+");
fseek(fp,pos,0);
```

```
fwrite(&anag, sizeof(struct per), 1, fp);
```

che consente di aggiornare il file a partire dalla posizione attuale del puntatore al file stabilita per mezzo della funzione di libreria `fseek()`.
La funzione `cer_per()`, invocata da `ric_per()`, effettua delle letture sul file per mezzo dell'istruzione

```
n = fread(&anag, sizeof(struct per), 1, fp);
```

alla ricerca della sequenza di byte che corrisponda a cognome, nome ed età della persona cercata.

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

#define DIM    31
#define MENU    0
#define INS     1
#define CAN     2
#define RIC     3
#define VIS     4
#define OUT   100

/* Semplice struttura che modella una persona */
struct per {
  char cognome[DIM];
  char nome[DIM];
  char ind[DIM];
  int  eta;
};

/* Puntatore al file */
FILE *fp;

/* La variabile di appoggio anag per le operazioni
sul file                                                    */
struct per anag;

int men_per(void);
void ins_per(void);
void ric_per(void);
void can_per(void);
long cer_per(char *, char *, int);
void eli_per(long pos);
void vis_per(void);
void vis_anagrafe(void);

/* Presenta il menu e lancia la funzione scelta */
void main()
{
int scelta = MENU;
while(scelta!=OUT) {
  switch(scelta) {
    case MENU:
      scelta = men_per();
      if(scelta == MENU)
         scelta = OUT;
      break;
    case INS:
      ins_per();
      scelta = MENU;
      break;
    case CAN:
```

```
      can_per();
      scelta = MENU;
      break;
    case RIC:
      ric_per();
      scelta = MENU;
      break;
  case VIS:
      vis_anagrafe();
      scelta = MENU;
      break;}
}
}

/* menu */
int men_per(void)
{
int scelta;
char invio;
int true = 1;

while(true){
  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
  printf("\t\t\t ANAGRAFE");
  printf("\n\n\n\t\t\t 1. Immissione Persona");
  printf("\n\n\t\t\t 2. Cancellazione Persona");
  printf("\n\n\t\t\t 3. Ricerca Persona");
  printf("\n\n\t\t\t 4. Visualizza anagrafe");
  printf("\n\n\t\t\t 0. Fine");
  printf("\n\n\n\t\t\t\t Scegliere una opzione: ");
  scanf("%d", &scelta);
  scanf("%c", &invio);
  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
  switch(scelta) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
      return(scelta);
    default:
      break;
  }
}
return(0);
}

/* inserisce persona nel file anag.dat */
void ins_per(void)
{
char invio;
/* Se il file non esiste lo crea,
   i dati sono appesi in fondo al file   */
fp = fopen("anag.dat","a+");

printf("\n\t\tINSERIMENTO PERSONA");
printf("\n\t\t---------------\n\n");
printf("\nCognome : ");
gets(anag.cognome);
printf("\nNome : ");
gets(anag.nome);
printf("\nIndirizzo : ");
gets(anag.ind);
```

```
printf("\nEtà : ");
scanf("%d", &anag.eta);
scanf("%c", &invio);

fwrite(&anag, sizeof(struct per), 1, fp);
fclose(fp);
}

/* Cancella una persona dall'anagrafe, se presente */
void can_per(void)
{
char pausa;
char cognome[DIM], nome[DIM];
int eta;
long pos;
printf("\n\t\tCANCELLA PERSONA");
printf("\n\t\t------\n\n");
printf("\nCognome : ");
gets(cognome);
printf("\nNome : ");
gets(nome);
printf("\nEtà : ");
scanf("%d", &eta);
scanf("%c", &pausa);
/* invoca ricerca persona */
pos = cer_per(cognome, nome, eta);
if(pos == -1) {
  printf("\nPersona non presente in anagrafe");
  scanf("%c", &pausa);
  return;
}
/* invoca visualizza persona */
vis_per();
printf("\nConfermi cancellazione ? (S/N) ");
scanf("%c", &pausa);
if(pausa=='S' || pausa=='s') {
  eli_per(pos);
  return;
}
}

/* Elimina persona dall'anagrafe */
void eli_per(long pos)
{
strcpy(anag.cognome, "");
strcpy(anag.nome, "");
strcpy(anag.ind, "");
anag.eta = 0;
fp = fopen("anag.dat","r+");
fseek(fp,pos,0);
fwrite(&anag, sizeof(struct per), 1, fp);
fclose(fp);
}

/* Ricerca persona */
void ric_per(void)
{
char pausa;
char cognome[DIM], nome[DIM];
int eta;
long pos;
/* Inserimento dati persona da ricercare */
printf("\n\t\tRICERCA PERSONA");
```

```
printf("\n\t\t------\n\n");
printf("\nCognome : ");
gets(cognome);
printf("\nNome : ");
gets(nome);
printf("\nEtà : ");
scanf("%d", &eta);
scanf("%c", &pausa);
/* Invoca la funzione di scansione sequenziale */
pos = cer_per(cognome, nome, eta);
if(pos == -1) {
  printf("\nPersona non presente in anagrafe");
  scanf("%c", &pausa);
  return;
}
vis_per();
scanf("%c", &pausa);
}

/* Effettua una scansione sequenziale del file anag.dat alla ricerca di una
persona che abbia determinati cognome, nome ed età */
long cer_per(char *cg, char *nm, int et)
{
int n;
long pos = 0L;

fp = fopen("anag.dat", "r");

for(;;) {
  n = fread(&anag, sizeof(struct per), 1, fp);
  if(n==0) {
    fclose(fp);
    pos = -1;
    return (pos);
  }
  else
    if(strcmp(cg, anag.cognome) == 0)
      if(strcmp(nm, anag.nome) == 0)
        if(et == anag.eta) {
            pos = ftell(fp);
              fclose(fp);
          return(pos-sizeof(struct per));
          }
  }
}

/* visualizza persona */
void vis_per(void)
{
printf("\n\n--------------------\n");
printf("\n\t\tCognome : %s", anag.cognome);
printf("\n\t\tNome : %s", anag.nome);
printf("\n\t\tIndirizzo : %s", anag.ind);
printf("\n\t\tEtà : %d", anag.eta);
printf("\n\n--------------------\n");
}

/* Visualizza l'anagrafe completa */
void vis_anagrafe(void)
{
int n; char pausa;
fp = fopen("anag.dat", "r");
do {
```

```
  n = fread(&anag, sizeof(struct per), 1, fp);
  if(n==0) fclose(fp);
  else {
    vis_per();
    scanf("%c", &pausa);
  }
}
while(n!=0);
}
```

Listato 13.7 Gestione anagrafica su file

13.6 Standard input e standard output

Quando un programma va in esecuzione il sistema apre automaticamente tre file pointer, mediante i quali è possibile scrivere messaggi a video e acquisire dati dalla tastiera. Questi tre file pointer prendono il nome di Standard Input (`stdin`), Standard Output (`stdout`) e Standard Error (`stderr`) e possono essere utilizzati dalle funzioni di accesso ai file descritte nei precedenti paragrafi.
Il file pointer `stdin` è associato dal sistema alla tastiera, i due file pointer `stdout` e `stderr` sono entrambi assegnati al video. Per scrivere un messaggio a video si può allora utilizzare, per esempio, la funzione `fprintf`:

```
#include <stdio.h>
main()
{
fprintf(stdout,"Ciao lettore\n");
}
```

Dunque le funzioni che abbiamo utilizzato per accettare valori da tastiera e mandare uscite al video corrispondono a usi particolari delle funzioni di uso generale esaminate in questo capitolo. Si hanno le seguenti equivalenze:

```
printf(...)    ->    fprintf(stdout,...)
scanf(...)     ->    fscanf(stdin,...)
getchar()      ->    fgetc(stdin)
putchar(c)     ->    fputc(c, stdout)
eof()          ->    feof(stdin)
```

La sintassi delle funzioni a sinistra è più sintetica, perché quelle sulla destra devono specificare che desiderano operare sullo standard input o sullo standard output.
Il programmatore deve prestare molta attenzione all'utilizzo delle due funzioni con sintassi abbreviata `gets` e `puts`, il cui comportamento è simile ma non uguale a quello delle funzioni `fgets` e `fputs`. Infatti `gets` legge una riga da tastiera ma elimina il carattere di newline e `puts` scrive una riga a video aggiungendo automaticamente un carattere di newline. Ne consegue che :

`gets(buf,n)` non equivale a `fgets(buf, n, stdin)`
`puts(buf)` non equivale a `fputs(buf, stdout)`