14.13 Gestione di una sequenza ordinata

In questo paragrafo considereremo il problema: inserire, eliminare valori interi da una lista lineare mantenendo la lista ordinata in modo crescente; visualizzare quindi la lista. Non si conosce a priori in quale sequenza e quante volte l'utente sceglierà di effettuare le operazioni sopra indicate. Il menu è il seguente.

```
GESTIONE DI UNA LISTA DI VALORI ORDINATI
        MEDIANTE UNA STRUTTURA A LISTA

        1. Per inserire un elemento
```

```
2. Per eliminare un elemento
3. Per visualizzare la lista
0. Per finire

  Scegliere una opzione:
```

Nel Listato 14.6 viene presentato il programma completo. In Figura 14.20 abbiamo invece l'inserimento e in Figura 14.21 l'eliminazione. Il lettore provi a disegnare una lista con valori ordinati e a eseguire manualmente passo passo le istruzioni delle funzioni.

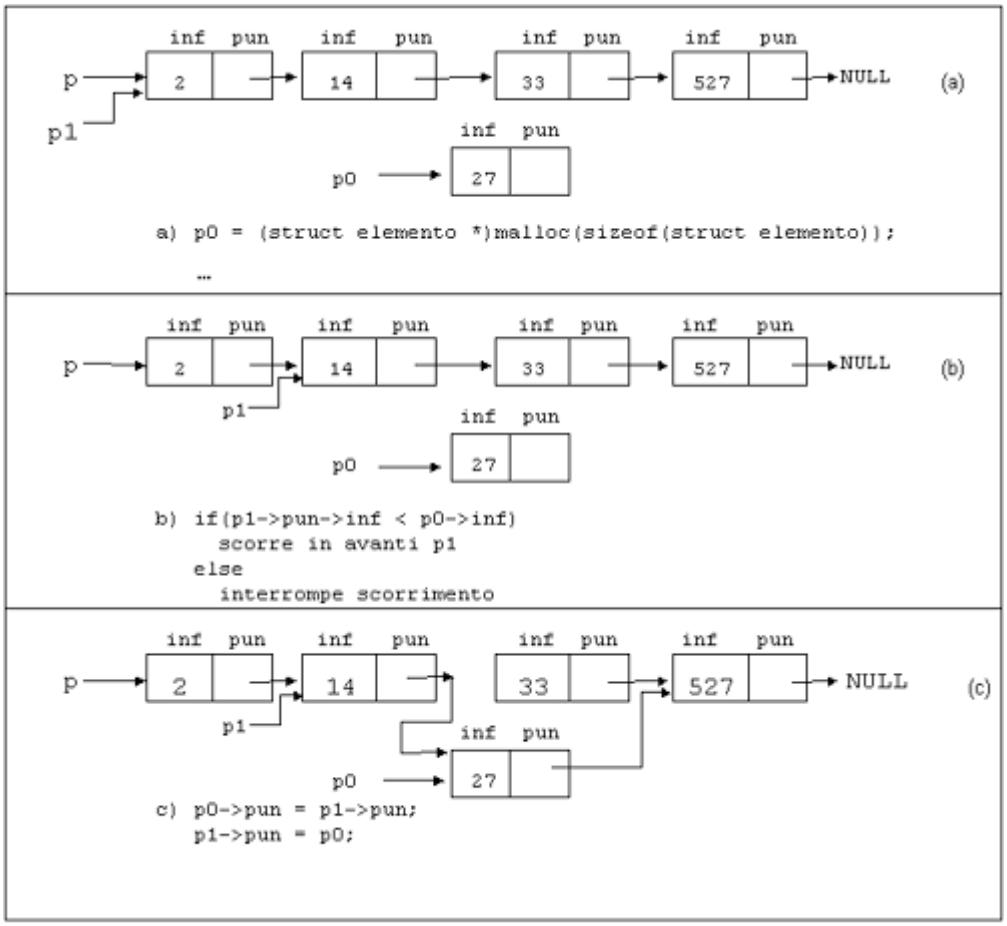

Figura 14.20 Inserimento di un elemento in una lista ordinata; nel caso preso in esame l'inserimento avviene in una posizione intermedia della lista

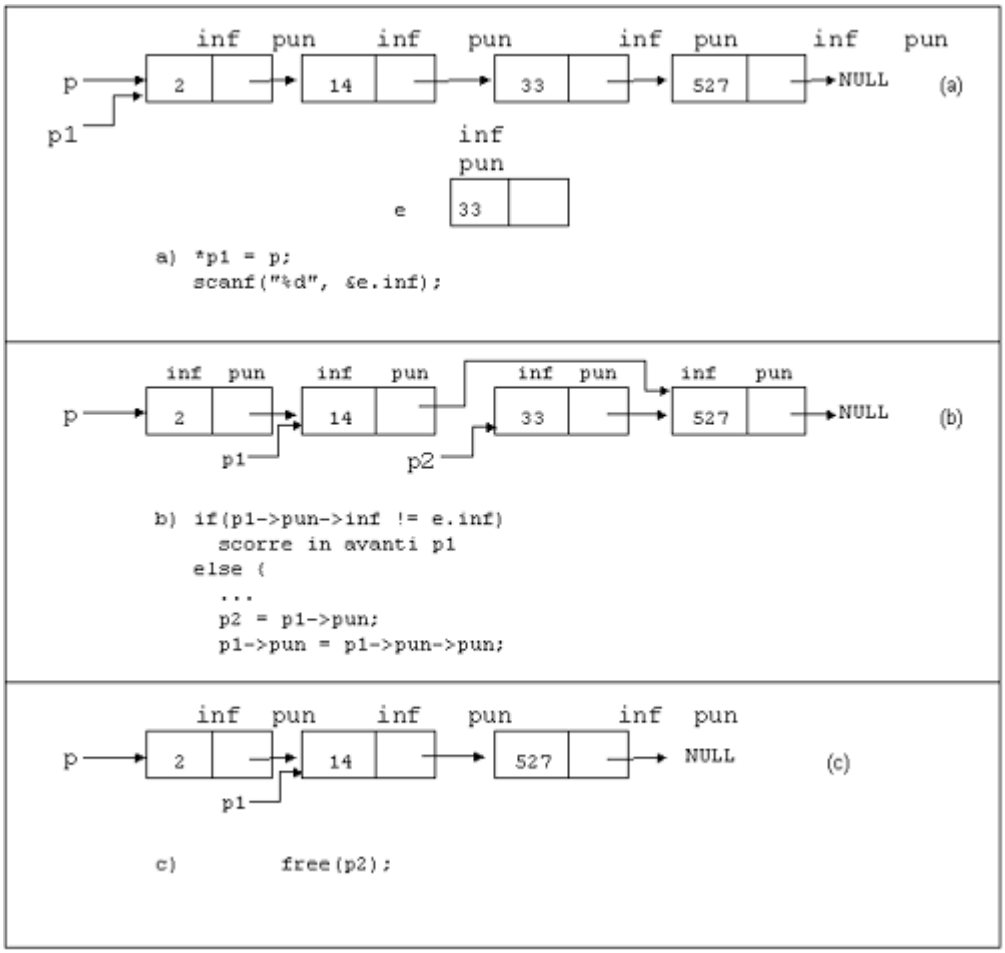

Figura 14.21 Eliminazione di un elemento da una lista ordinata; nel caso in esame l'eliminazione avviene in una posizione intermedia della lista

```
/* GESTIONE DI LISTA ORDINATA
   Operazioni di inserimento, eliminazione e visualizzazione
   Utilizza una lista lineare per implementare la pila        */

#include <stdio.h>
#include<malloc.h>

struct elemento {
  int inf;
  struct elemento *pun;
};

void gestione_lista(void);
struct elemento *inserimento(struct elemento *);
struct elemento *eliminazione(struct elemento *);
void visualizzazione(struct elemento *);

main()
{
  gestione_lista();
}


void gestione_lista(void)
{
```

```
struct elemento *punt_lista = NULL;
int scelta = -1;
char pausa;

while(scelta!=0) {
  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");
  printf("\t\tGESTIONE DI UNA SEQUENZA DI VALORI ORDINATI\n");
  printf("\t\t     MEDIANTE UNA STRUTTURA A LISTA");
  printf("\n\n\n\t\t\t 1. Per inserire un elemento");
  printf("\n\n\t\t\t 2. Per eliminare un elemento");
  printf("\n\n\t\t\t 3. Per visualizzare la lista");
  printf("\n\n\t\t\t 0. Per finire");
  printf("\n\n\n\t\t\t\t Scegliere una opzione: ");
  scanf("%d", &scelta);
  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");

  switch(scelta) {
     case 1:
        punt_lista = inserimento(punt_lista);
        break;
     case 2:
        punt_lista = eliminazione(punt_lista);
        break;
     case 3:
        visualizzazione(punt_lista);
        printf("\n\nQualsiasi tasto per continuare...");
        scanf("%c%c", &pausa, &pausa);
        break;
  }
}
}


/* Visualizzazione della lista */
void visualizzazione(struct elemento *p)
{
struct elemento *paus = p;

printf("\npunt_lista---> ");
while(paus!=NULL) {
  printf("%d---> ", paus->inf);
  paus = paus->pun;
}
printf("NULL");
}


/* Inserimento del valore passato dall'utente nella lista
   mantenendo l'ordinamento                                */
struct elemento *inserimento(struct elemento *p)
{
struct elemento *p0, *p1;
int posizione;

/* Creazione elemento */
p0 = (struct elemento *)malloc(sizeof(struct elemento));

printf("\nInserisci l'informazione (un numero intero): ");
scanf("%d", &p0->inf);
```

```
if(p==NULL) {           /* Se la lista è vuota, l'elemento      */
  p = p0;               /* diventa il primo e unico della lista */
  p->pun = NULL;
}
else {
  if(p->inf > p0->inf) {    /* Se il valore dell'elemento è  */
     p0->pun = p;           /* inferiore al primo l'elemento */
     p = p0;                /* diventa il primo della lista  */
  }
  else {           /* Ricerca della posizione di inserimento */
     p1 = p;
     posizione = 0;
     while(p1->pun!=NULL && posizione!=1) {
       if(p1->pun->inf < p0->inf)
         p1 = p1->pun;          /* Scorre in avanti p1       */
       else
         posizione = 1;         /* Interrompe lo scorrimento */
     }
     p0->pun = p1->pun; /* Connessione all'elemento successivo */
     p1->pun = p0;     /* Connessione dall'elemento precedente */
  }
}
return(p);    /* Ritorno del puntatore all'inizio della lista */
}


/* Eliminazione dell'elemento richiesto dalla lista ordinata */
struct elemento *eliminazione(struct elemento *p)
{
struct elemento *p1 = p, *p2;
struct elemento e;
int posizione = 0;
char pausa;

printf("\nInserisci l'informazione da eliminare: ");
scanf("%d", &e.inf);

if(p1!=NULL) {              /* Se la lista è vuota fine */
  if(p1->inf == e.inf) {    /* Se è il primo da eliminare */
    p2 = p1;
    p = p->pun;       /* si modifica il puntatore alla testa  */
    free(p2);
    return(p);
  }
  else {               /* Ricerca dell'elemento da eliminare */
    while(p1->pun!=NULL && posizione!=1) {
       if(p1->pun->inf!=e.inf)
          p1 = p1->pun;          /* Scorre in avanti p1 */
       else {
          posizione = 1;         /* Interrompe lo scorrimento */
          p2 = p1->pun;
          p1->pun = p1->pun->pun;    /* Eliminazione elemento */
          free(p2);                  /* Libera la memoria */
        return( p );
       }
    }
  }
}
```

```
if(!posizione) {
  printf("\nElemento non incontrato nella lista ");
  scanf("%c%c", &pausa, &pausa);
}
return(p);
}
```

Listato 14.6 Gestione di una lista ordinata