```c
/* un terzo esempio di uso di liste: qui creo un albero binario. */
/* uso una funzione ricorsiva */

#include <stdio.h>
#include <math.h>

#define MY_MAX 100

#define SUCCESSO 0
#define ROVINA -9

FILE *f_input;
char *f_nome_input= "leopardi2.dat";

int my_end;

void my_start(void);
void apri_file(void);
char *leggi_una_parola(void);

struct lista_parole{
  char *p_stringa;
  struct lista_parole *prossima_parola_sinistra;
  struct lista_parole *prossima_parola_destra;
};

struct lista_parole *aggiungi_all_albero(struct lista_parole *, char *);
void stampa_albero_in_ordine_alfabetico(struct lista_parole *);

/*****************************************************************************/
int main(void)
{
  int i;
/* creo la lista e visto che all'inizio e' vuoto la inizializzo a NULL */
  struct lista_parole *radice_albero=NULL;
  char *p_scratch;


  my_start();
  apri_file();

  my_end=0;
  for(i=0;;i++){
    p_scratch = leggi_una_parola();
    radice_albero = aggiungi_all_albero(radice_albero,p_scratch);
    if(my_end==1)break;
  }
  fclose(f_input);
  stampa_albero_in_ordine_alfabetico(radice_albero);
}

/*****************************************************************************/
void my_start(void)
{
  printf("# Costruiremo un albero binario\n");
}

/*****************************************************************************/
void apri_file(void)
{
  f_input = fopen(f_nome_input,"r");
  if(f_input==NULL){
    printf("abort: impossibile aprire il file di input in lettura: %s\n",
           f_nome_input);
```

```
    exit(-9);
  }
  printf("# E' stato aperto il file di input: %s\n",
         f_nome_input);
}

/***************************************************************************/
char *leggi_una_parola(void)
{
  int j;
  char my_string[MY_MAX];
  char * p_interno;

  for(j=0; j<MY_MAX; j++){
    my_string[j] = fgetc(f_input);
    if(my_string[j]=='\n'){j--; continue;}/*per ignorare il fine riga*/
    if(my_string[j]==EOF){
      my_end=1;
      my_string[j]='\0';
      break;}
    if(my_string[j]==' '){
      my_string[j]='\0';
      break;
    }
  }
  /* per sicurezza mi sto riservando un carattere in piu' oltre
    al terminatore \0 */
  if(j>=MY_MAX-1){
    printf("Interruzione del programma: la parola era troppo lunga.\n");
    printf("Ricompilare con un valore di MY_MAX piu grande, era %d.\n",
           MY_MAX);
    exit(-9);
  }
  p_interno = (char *)malloc(strlen(my_string)+1);
  if(p_interno==NULL){
    printf("Interruzione del programma: fallita malloc 1 in c_l_i\n");
    exit(-9);
  }
  strcpy(p_interno,my_string);

  /* printf("Parola di lunghezza %d: %s\n",strlen(p_interno),p_interno); */

  return p_interno;
}

/***************************************************************************/
struct lista_parole* aggiungi_all_albero(struct lista_parole* lista_input,
                                         char *p_scratch)
{
  int differenza_stringa;
  if(lista_input==NULL){
    /* qui posso aggiungere */
    /* printf("Aggiungo dato che sono al NULL\n");fflush(stdout); */
    lista_input =
    (struct lista_parole *) malloc(sizeof(struct lista_parole));
    if(lista_input==NULL){
      printf("Interruzione del programma: fallita malloc 1 in a_a_a\n");
      exit(-9);
    }
    lista_input->p_stringa = p_scratch;
    lista_input->prossima_parola_sinistra = NULL;
    lista_input->prossima_parola_destra = NULL;
  } else if((differenza_stringa =
            strcmp(lista_input->p_stringa,p_scratch))==0){
    /* notate che la parentesi (a=b)==c nella riga precedente e' cruciale! */
```

```
    /* printf("Questa parola era gia' nel dizionario\n");fflush(stdout); */
  } else if(differenza_stringa>0) {
    /* printf("Vado a sinistra\n");fflush(stdout); */
    lista_input->prossima_parola_sinistra =
      aggiungi_all_albero(lista_input->prossima_parola_sinistra,p_scratch);
  } else {
    /* printf("Vado a destra\n");fflush(stdout); */
    lista_input->prossima_parola_destra =
      aggiungi_all_albero(lista_input->prossima_parola_destra,p_scratch);
  }
  return lista_input;
}

/****************************************************************************/
void stampa_albero_in_ordine_alfabetico(struct lista_parole* lista_input)
{
  /* la ricorsione sembra un miracolo, a volte.. */
  /* stampa la parte dell'albero a sinistra, poi la parola stessa */
  /* poi la parte dell'albero a destra */
  if(lista_input!=NULL){
    stampa_albero_in_ordine_alfabetico(lista_input->prossima_parola_sinistra);
    printf("%s\n",lista_input->p_stringa);
    stampa_albero_in_ordine_alfabetico(lista_input->prossima_parola_destra);
  }
}
```