# 15.3 Visita in ordine simmetrico

Il problema di questo paragrafo è: ampliare il programma del paragrafo precedente in modo che venga effettuata anche la visita in ordine simmetrico e inoltre sia ricercato nell'albero un valore richiesto all'utente. Nel Listato 15.2 sono riportate le modifiche da apportare al programma e le nuove funzioni.
La funzione di visita `simmetrico` è analoga ad `anticipato` esaminata precedentemente. La differenza sta nel fatto che prima viene visitato il sottoalbero sinistro, poi viene visitata la radice e infine viene visitato il sottoalbero destro.
La variabile `trovato` del `main` è di tipo puntatore a una struttura `nodo`:

```
struct nodo *trovato;
```

Essa viene passata per indirizzo alla funzione `ricerca` che vi immette, se reperito, il puntatore al nodo che contiene il valore ricercato. La funzione `ricerca` si comporta come la visita in ordine anticipato: prima verifica se l'elemento ricercato è nella posizione corrente, poi lo ricerca nel sottoalbero sinistro, infine lo ricerca nel sottoalbero destro.

```
/* Da aggiungere alle dichiarazioni iniziali del Listato 15.1 */

void simmetrico(struct nodo *);
void ricerca(struct nodo *, int, struct nodo **);

/* Da aggiungere al main del Listato 15.1 */
```

```
struct nodo *trovato;
int chi;
...
printf("\nVISITA IN ORDINE SIMMETRICO\n");
simmetrico(radice);

printf("\nInserisci il valore da ricercare: ");
scanf("%d", &chi);

printf("\nRICERCA COMPLETA");
trovato = NULL;
ricerca(radice, chi, &trovato);
if(trovato != NULL)
  printf("\n Elemento %d presente \n", trovato->inf);
else
  printf("\n Elemento non presente\n");


/* Funzione che visita l'albero binario in ordine simmetrico.
   Da aggiungere al Listato 15.1                              */

void simmetrico(struct nodo *p)
{
if(p!=NULL) {
  simmetrico(p->alb_sin);
  printf("%d  ", p->inf);
  simmetrico(p->alb_des);
}
}


/* Funzione che ricerca un'etichetta nell'albero binario.
   Da aggiungere al Listato 15.1.
   Visita l'albero in ordine anticipato                    */

void ricerca(struct nodo *p, int val, struct nodo **p_ele)
{
if(p!=NULL)
  if(val == p->inf)      /* La ricerca ha dato esito positivo */
    *p_ele = p;          /* p_ele è passato per indirizzo
                            per cui l'assegnamento di p
                            avviene sul parametro attuale */
  else {
    ricerca(p->alb_sin, val, p_ele);
    ricerca(p->alb_des, val, p_ele);
}
}
```

Listato 15.2 Visita in ordine simmetrico e ricerca di un valore nell’albero binario