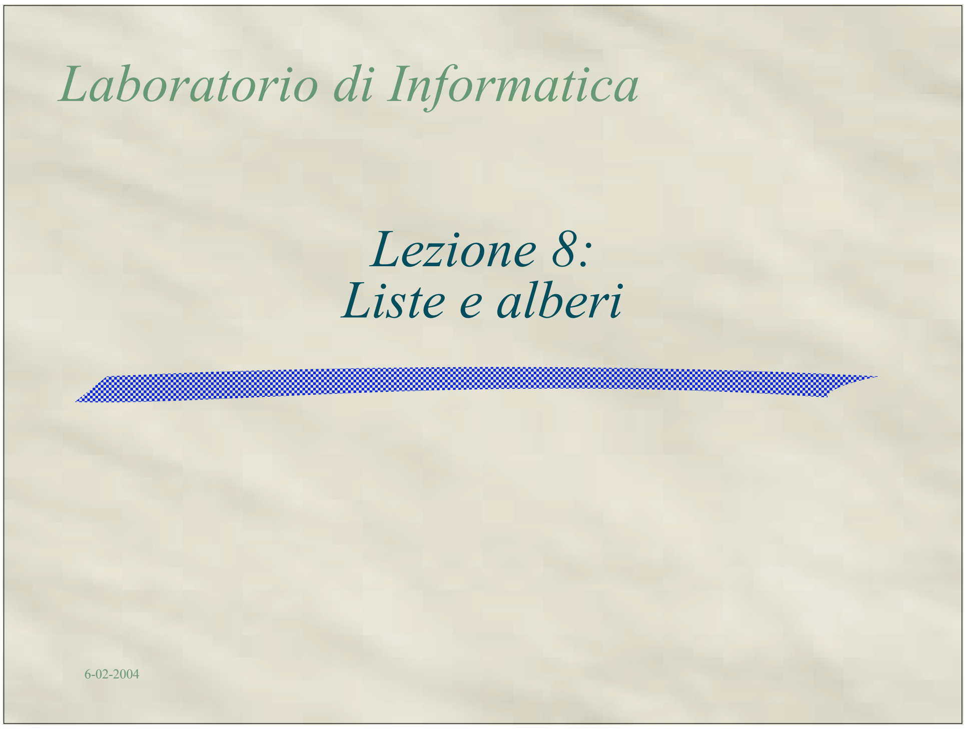
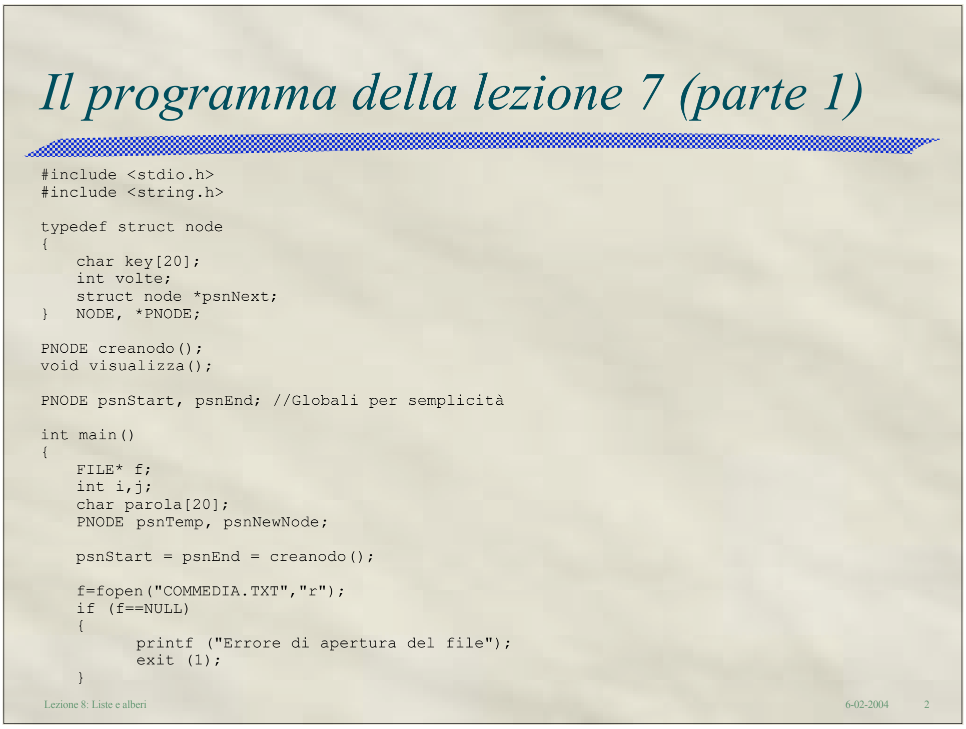


Laboratorio di Informatica

Lezione 8: Liste e alberi

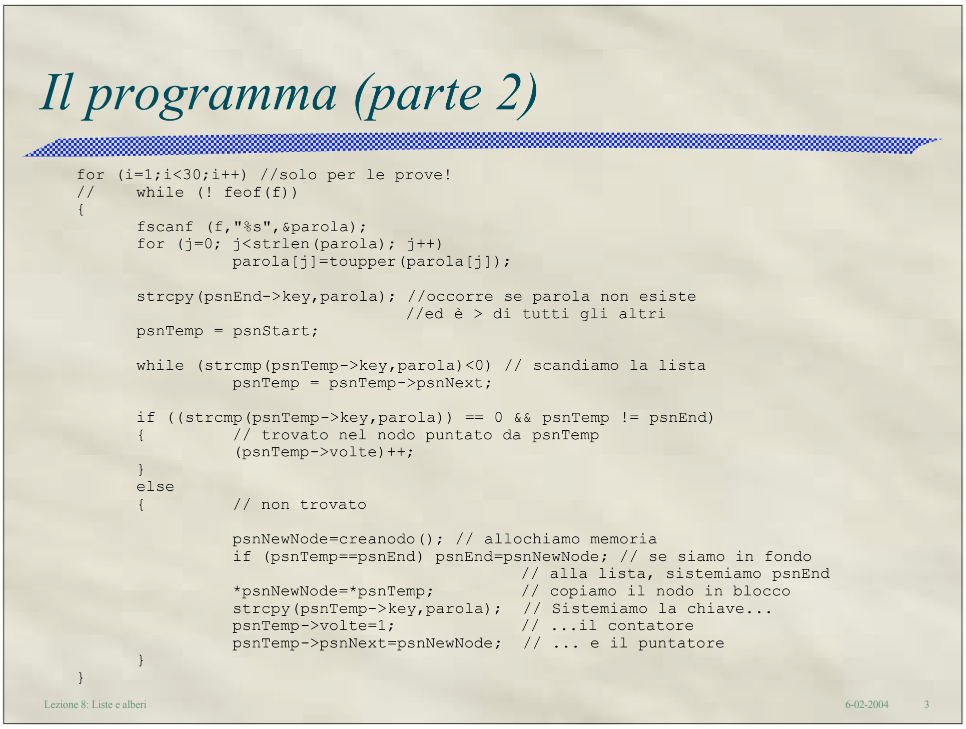
6-02-2004



Il programma della lezione 7 (parte 1)

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

typedef struct node
{
    char key[20];
    int volte;
    struct node *psnNext;
}   NODE, *PNODE;

PNODE creanodo();
void visualizza();

PNODE psnStart, psnEnd; //Globali per semplicità

int main()
{
    FILE* f;
    int i,j;
    char parola[20];
    PNODE psnTemp, psnNewNode;

    psnStart = psnEnd = creanodo();

    f=fopen("COMMEDIA.TXT","r");
    if (f==NULL)
    {
          printf ("Errore di apertura del file");
          exit (1);
    }
```

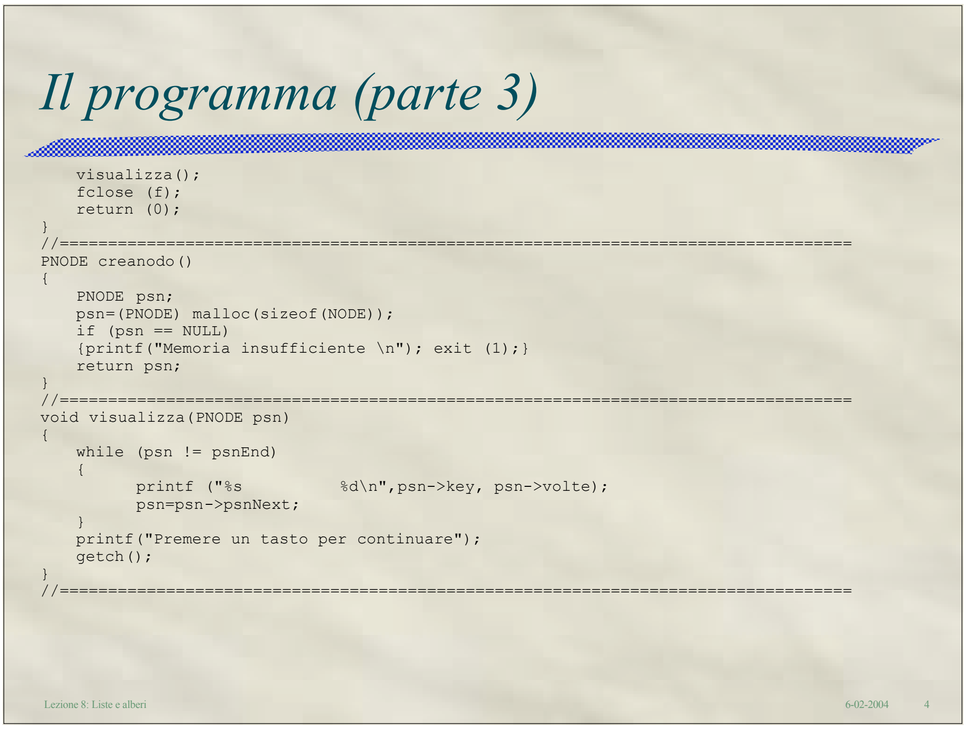
Lezione 8: Liste e alberi 6-02-2004 2



Il programma (parte 2)

```
    for (i=1;i<30;i++) //solo per le prove!
    //    while (! feof(f))
    {
          fscanf (f,"%s",&parola);
          for (j=0; j<strlen(parola); j++)
                    parola[j]=toupper(parola[j]);

          strcpy(psnEnd->key,parola); //occorre se parola non esiste
                                      //ed è > di tutti gli altri
          psnTemp = psnStart;

          while (strcmp(psnTemp->key,parola)<0) // scandiamo la lista
                    psnTemp = psnTemp->psnNext;

          if ((strcmp(psnTemp->key,parola)) == 0 && psnTemp != psnEnd)
          {         // trovato nel nodo puntato da psnTemp
                    (psnTemp->volte)++;
          }
          else
          {         // non trovato

                    psnNewNode=creanodo(); // allochiamo memoria
                    if (psnTemp==psnEnd) psnEnd=psnNewNode; // se siamo in fondo
                                                  // alla lista, sistemiamo psnEnd
                    *psnNewNode=*psnTemp;         // copiamo il nodo in blocco
                    strcpy(psnTemp->key,parola);  // Sistemiamo la chiave...
                    psnTemp->volte=1;             // ...il contatore
                    psnTemp->psnNext=psnNewNode;  // ... e il puntatore
          }
    }
```

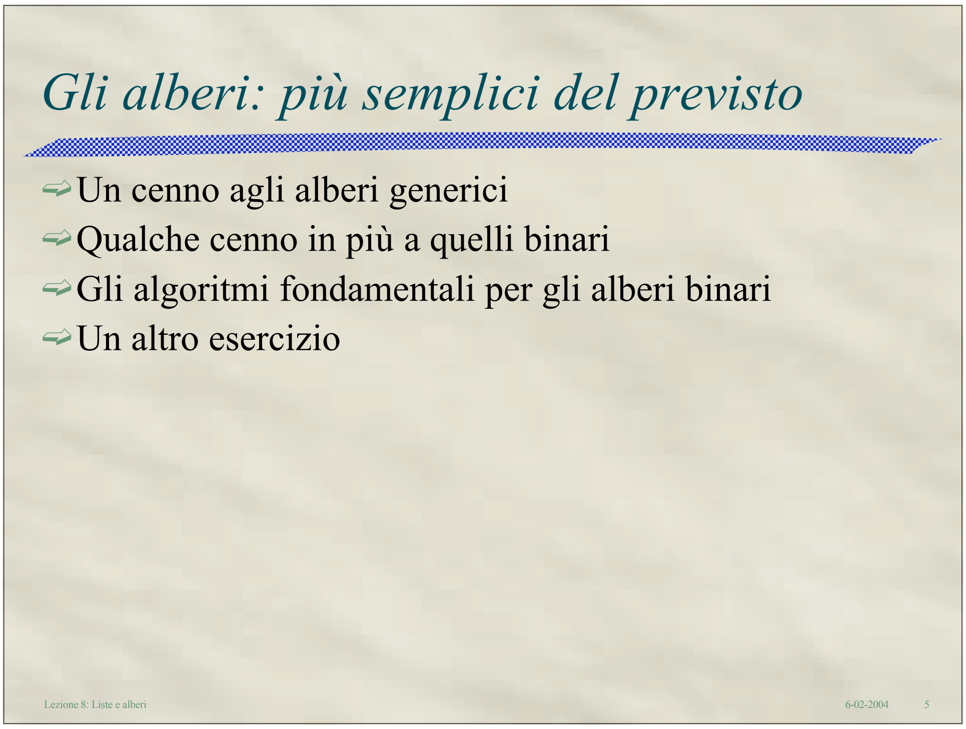
Lezione 8: Liste e alberi   6-02-2004   3



Il programma (parte 3)

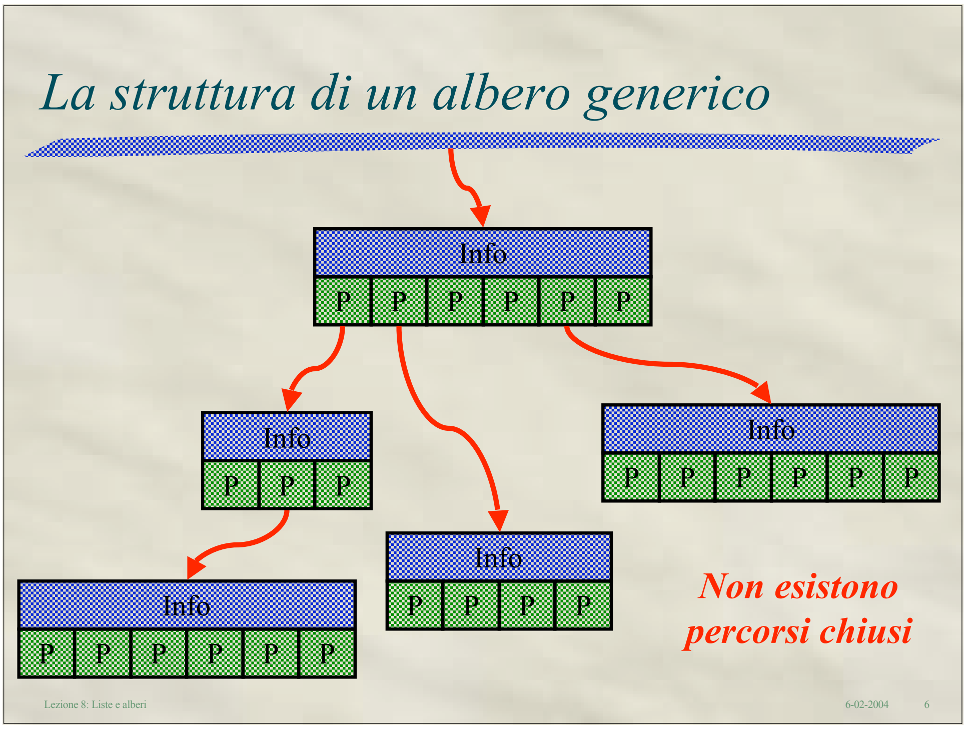
```
    visualizza();
    fclose (f);
    return (0);
}
//===================================================================================
PNODE creanodo()
{
    PNODE psn;
    psn=(PNODE) malloc(sizeof(NODE));
    if (psn == NULL)
    {printf("Memoria insufficiente \n"); exit (1);}
    return psn;
}
//===================================================================================
void visualizza(PNODE psn)
{
    while (psn != psnEnd)
    {
          printf ("%s          %d\n",psn->key, psn->volte);
          psn=psn->psnNext;
    }
    printf("Premere un tasto per continuare");
    getch();
}
//===================================================================================
```

Lezione 8: Liste e alberi   6-02-2004   4



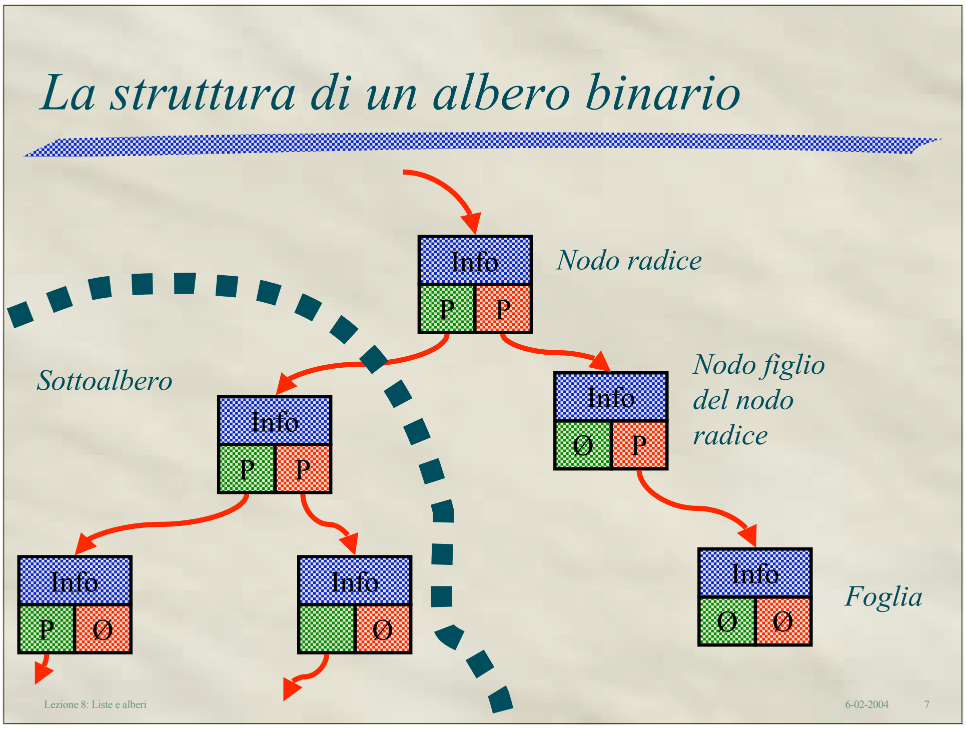
Gli alberi: più semplici del previsto

- Un cenno agli alberi generici
- Qualche cenno in più a quelli binari
- Gli algoritmi fondamentali per gli alberi binari
- Un altro esercizio

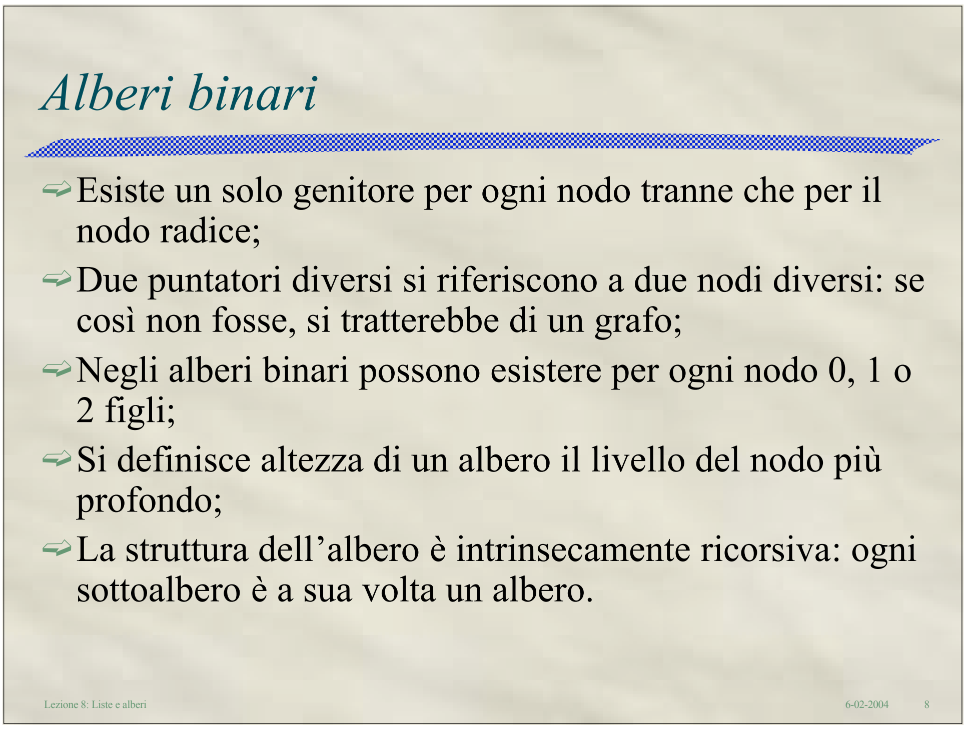


La struttura di un albero generico

Non esistono percorsi chiusi

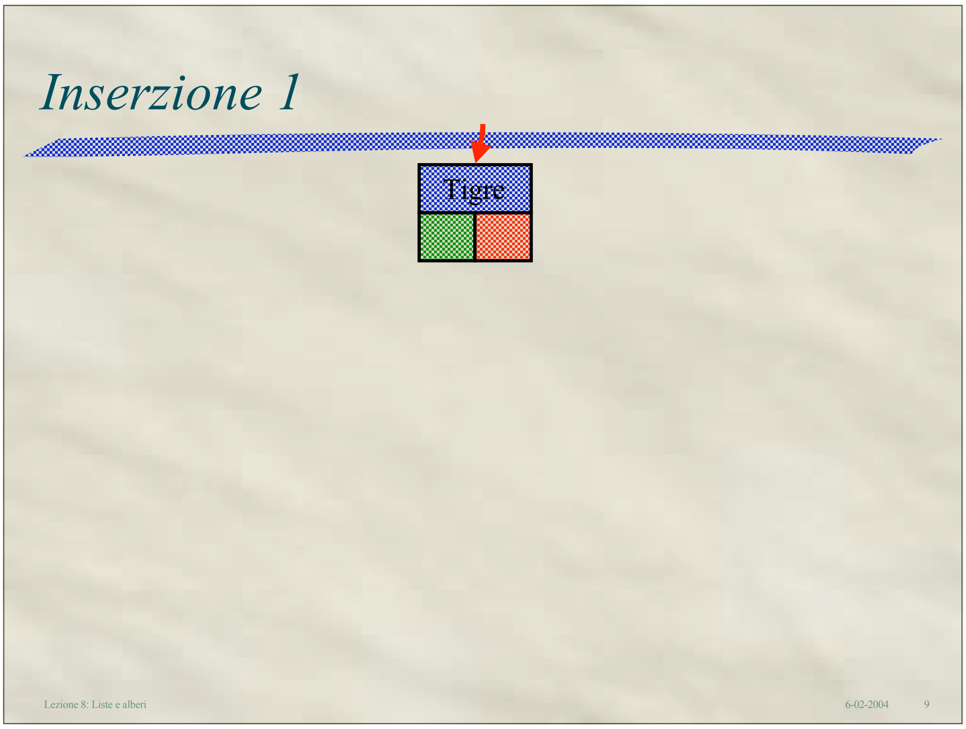


La struttura di un albero binario

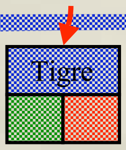


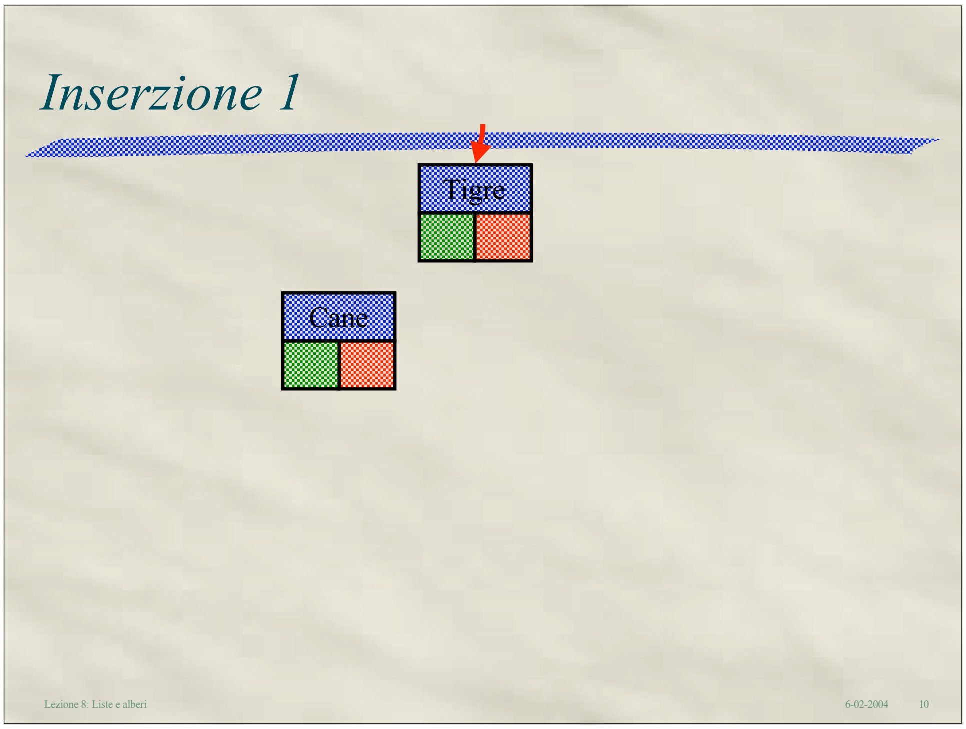
Alberi binari

- Esiste un solo genitore per ogni nodo tranne che per il nodo radice;
- Due puntatori diversi si riferiscono a due nodi diversi: se così non fosse, si tratterebbe di un grafo;
- Negli alberi binari possono esistere per ogni nodo 0, 1 o 2 figli;
- Si definisce altezza di un albero il livello del nodo più profondo;
- La struttura dell'albero è intrinsecamente ricorsiva: ogni sottoalbero è a sua volta un albero.

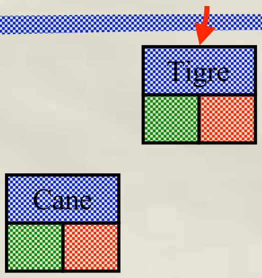


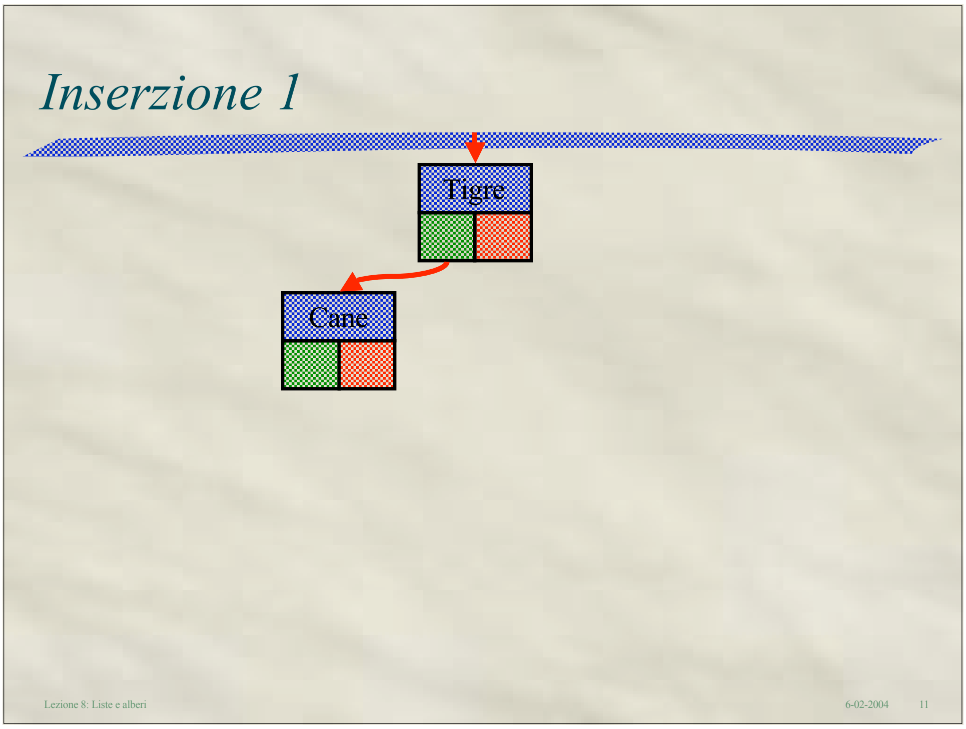
Inserzione 1



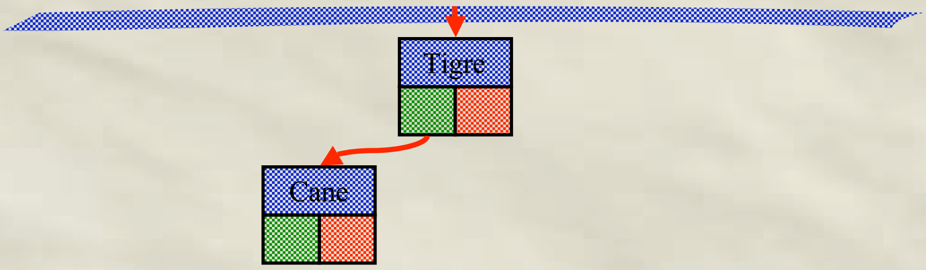


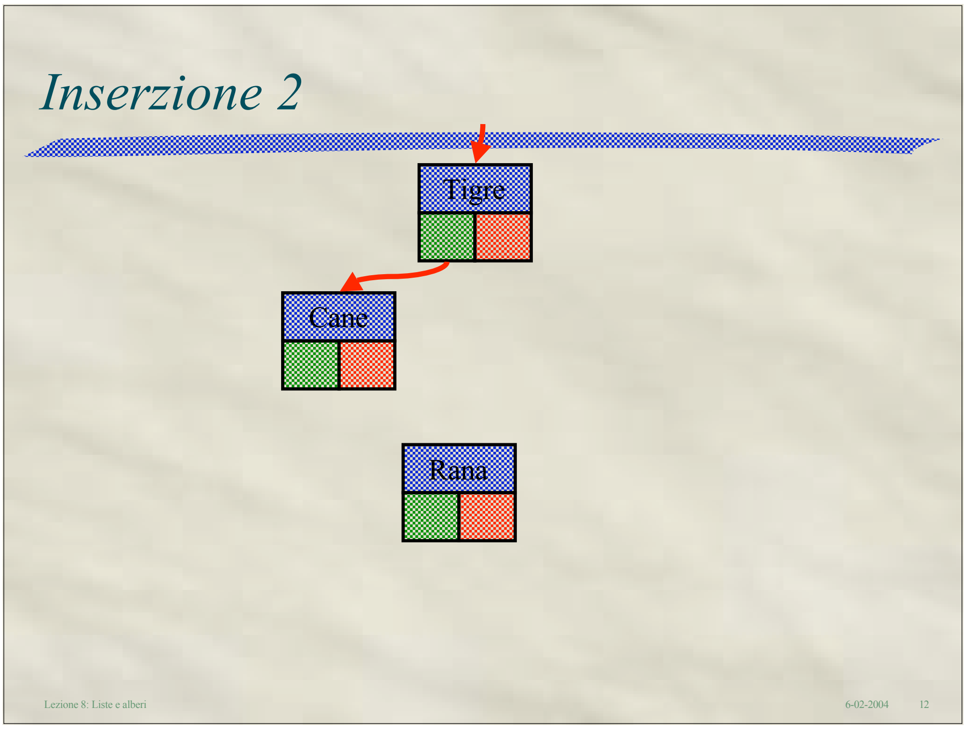
Inserzione 1



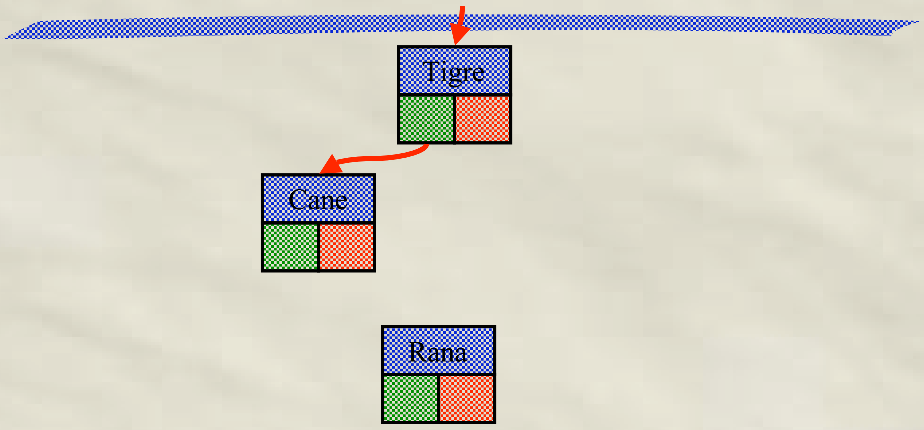


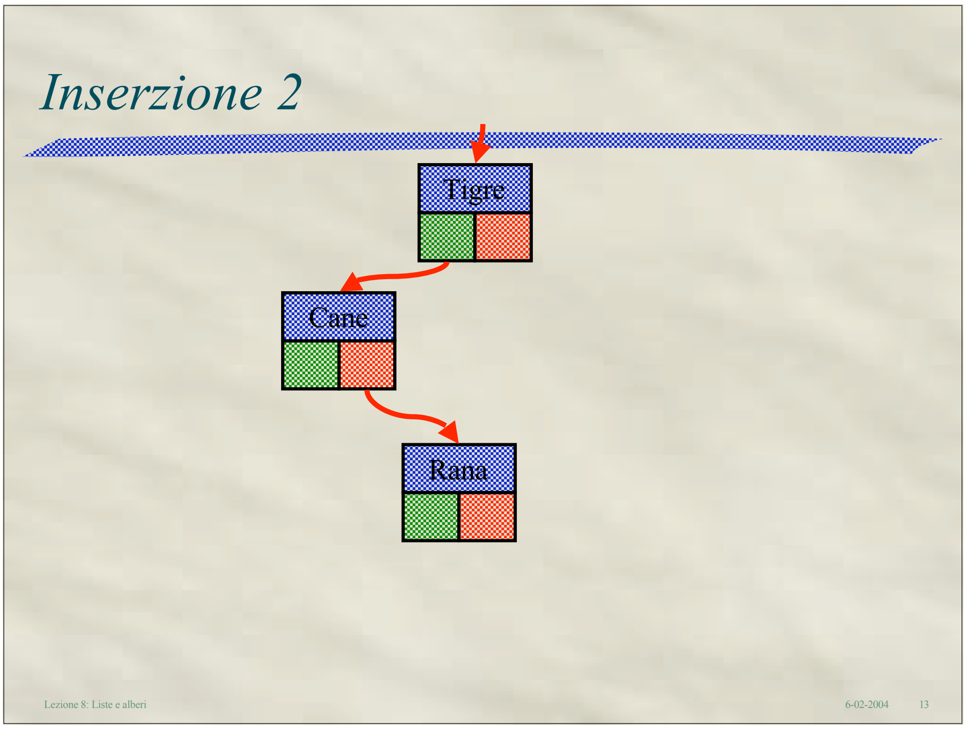
Inserzione 1



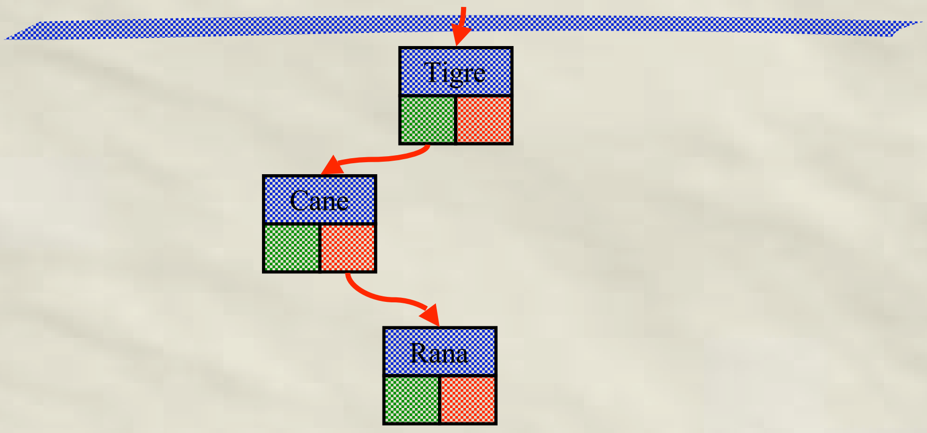


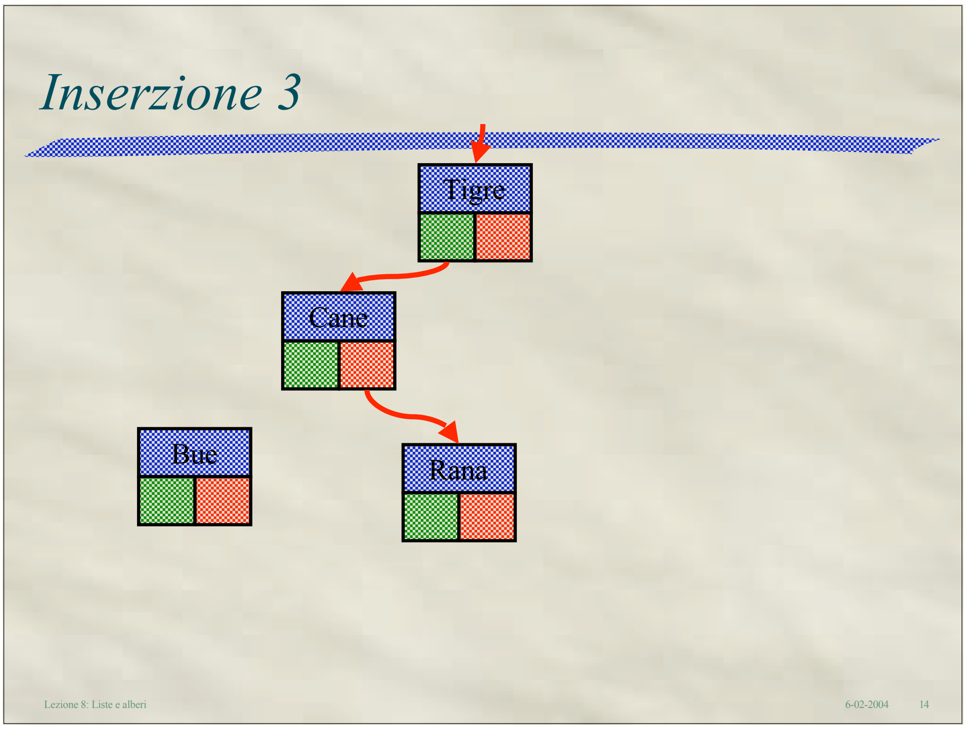
Inserzione 2



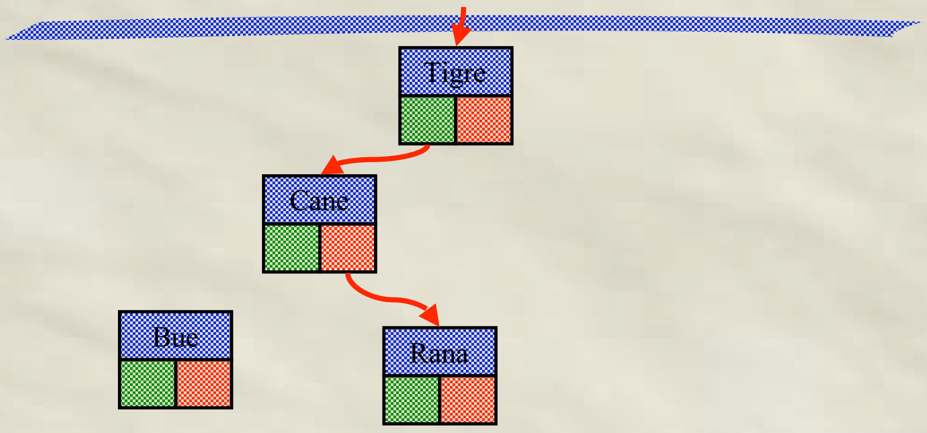


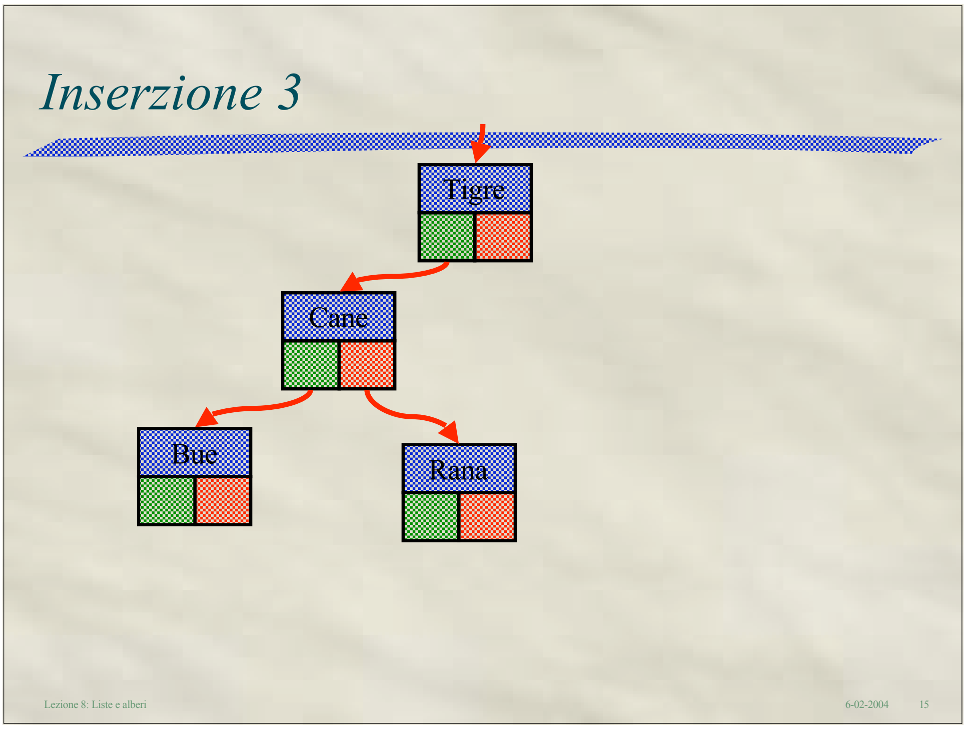
Inserzione 2



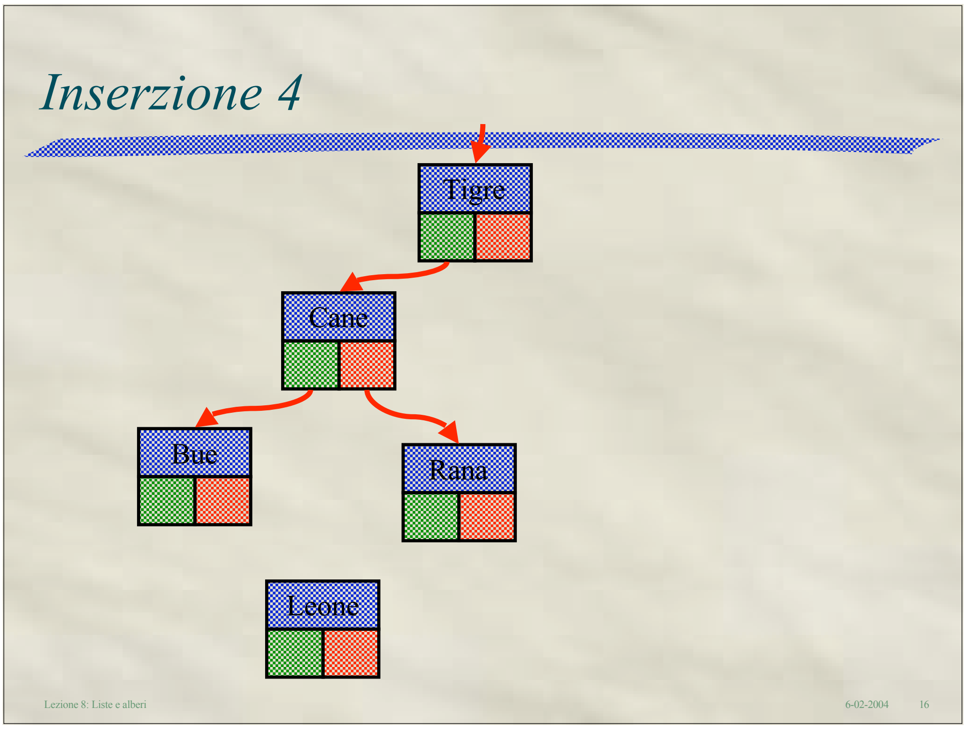


Inserzione 3

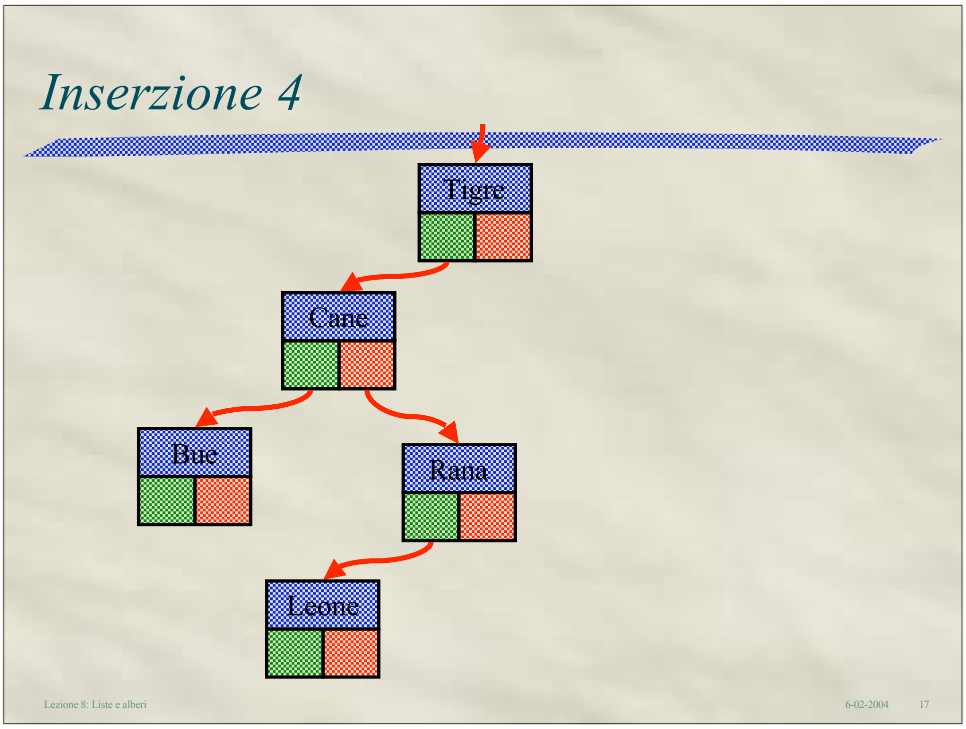




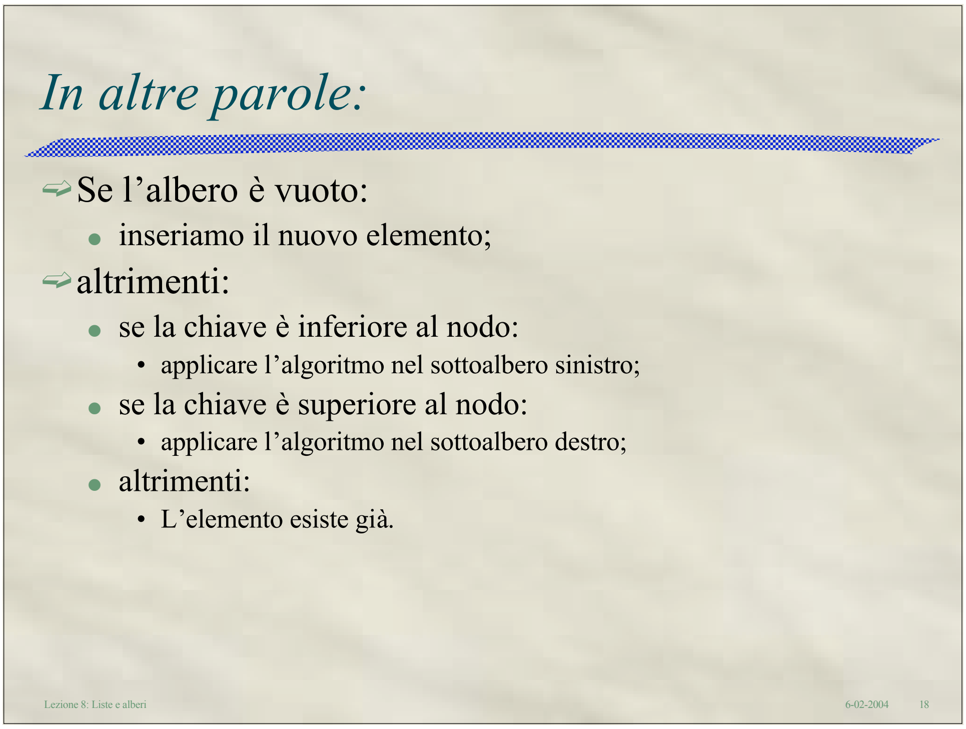
Inserzione 3



Inserzione 4

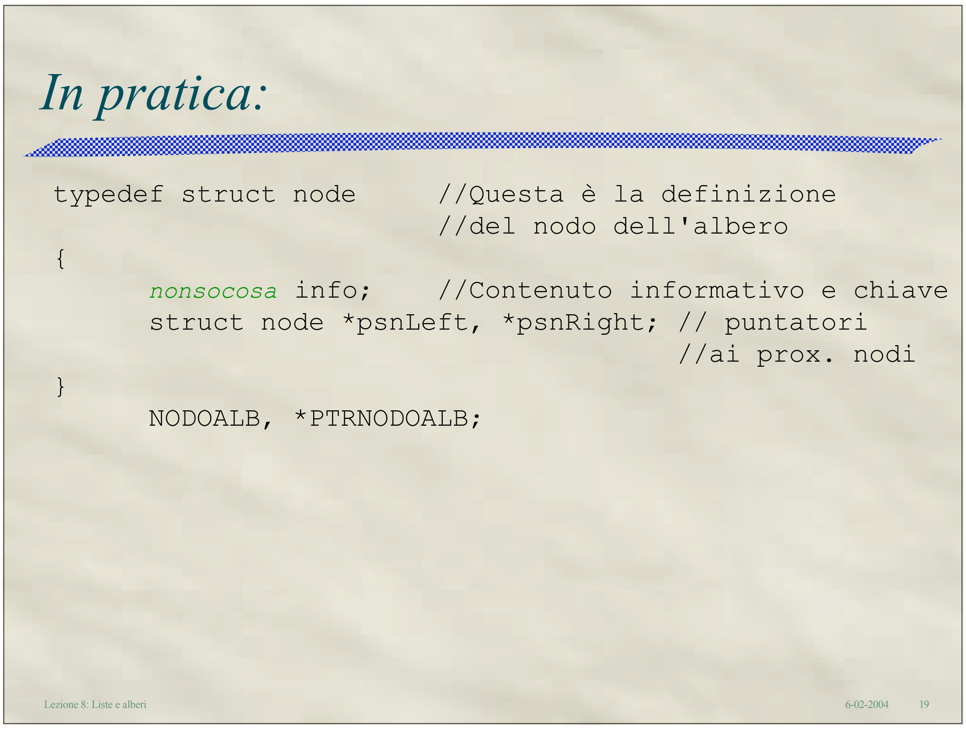


Inserzione 4



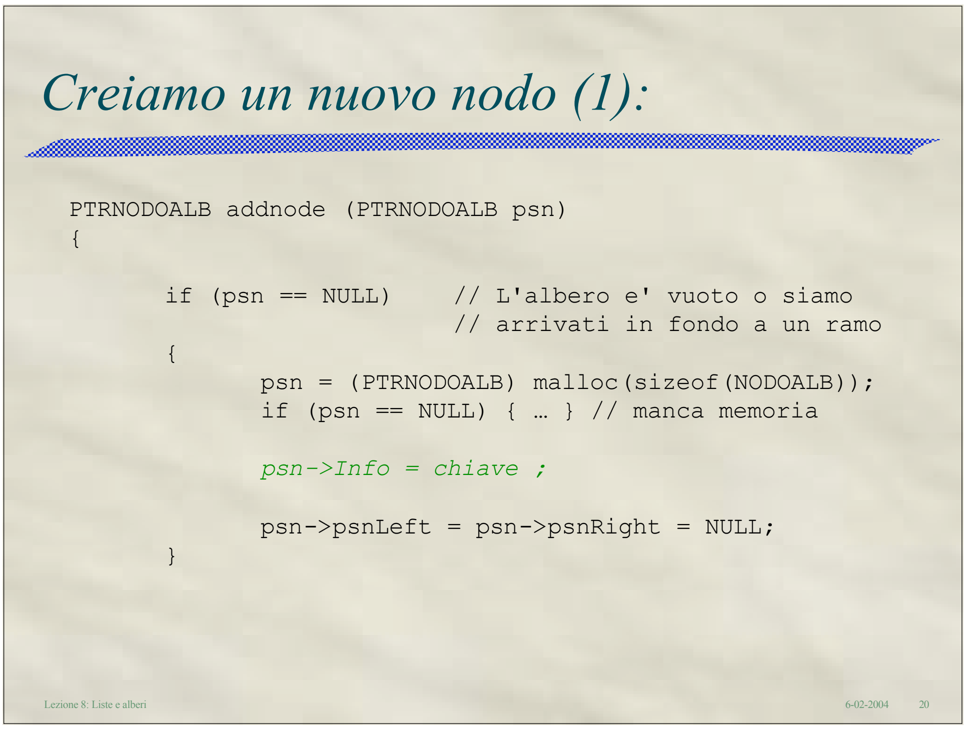
In altre parole:

- Se l’albero è vuoto:
  - inseriamo il nuovo elemento;
- altrimenti:
  - se la chiave è inferiore al nodo:
    - applicare l’algoritmo nel sottoalbero sinistro;
  - se la chiave è superiore al nodo:
    - applicare l’algoritmo nel sottoalbero destro;
  - altrimenti:
    - L’elemento esiste già.



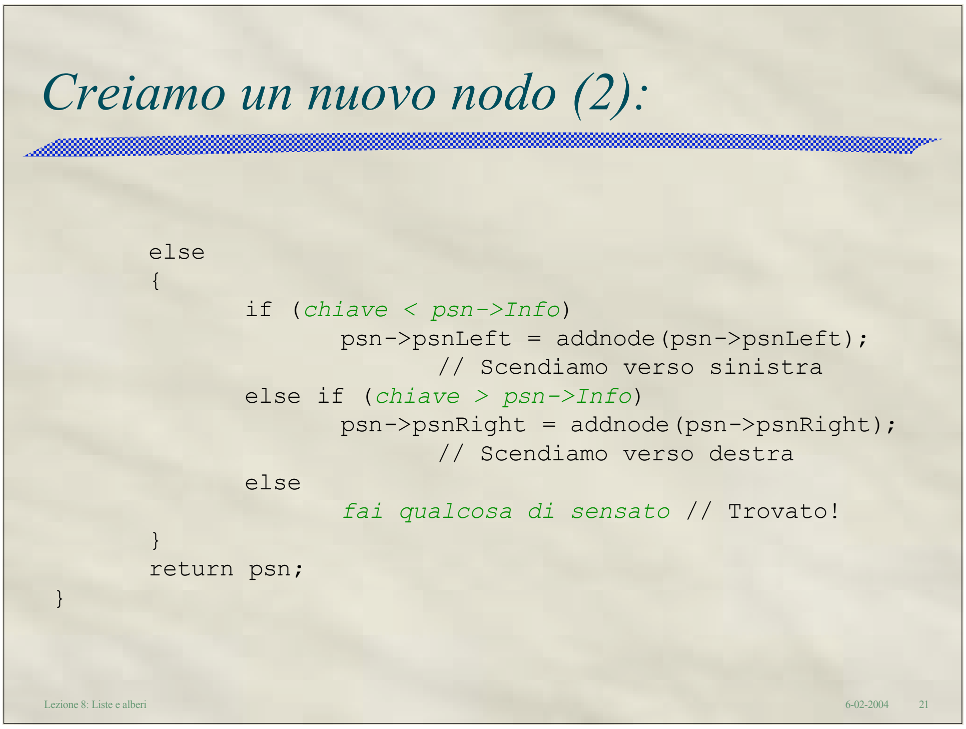
In pratica:

```
typedef struct node     //Questa è la definizione
                        //del nodo dell'albero
{
      nonsocosa info;   //Contenuto informativo e chiave
      struct node *psnLeft, *psnRight; // puntatori
                                       //ai prox. nodi
}
      NODOALB, *PTRNODOALB;
```



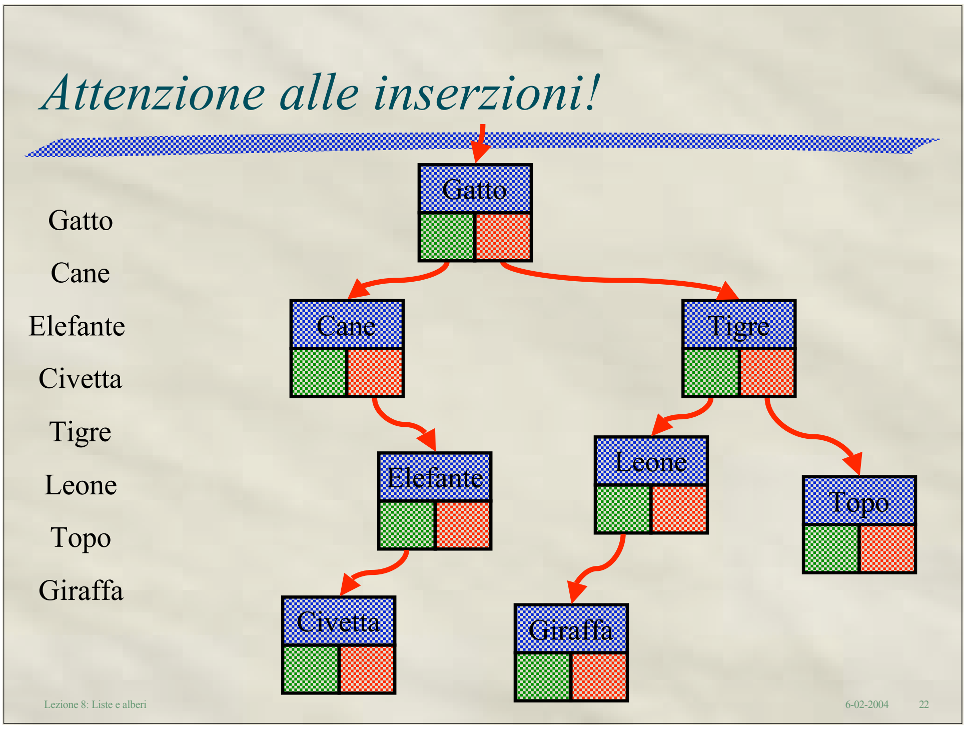
Creiamo un nuovo nodo (1):

```
PTRNODOALB addnode (PTRNODOALB psn)
{

      if (psn == NULL)    // L'albero e' vuoto o siamo
                          // arrivati in fondo a un ramo
      {
            psn = (PTRNODOALB) malloc(sizeof(NODOALB));
            if (psn == NULL) { … } // manca memoria

            psn->Info = chiave ;

            psn->psnLeft = psn->psnRight = NULL;
      }
```



Creiamo un nuovo nodo (2):

```
      else
      {
            if (chiave < psn->Info)
                  psn->psnLeft = addnode(psn->psnLeft);
                        // Scendiamo verso sinistra
            else if (chiave > psn->Info)
                  psn->psnRight = addnode(psn->psnRight);
                        // Scendiamo verso destra
            else
                  fai qualcosa di sensato // Trovato!
      }
      return psn;
}
```



Attenzione alle inserzioni!

Gatto

Cane

Elefante

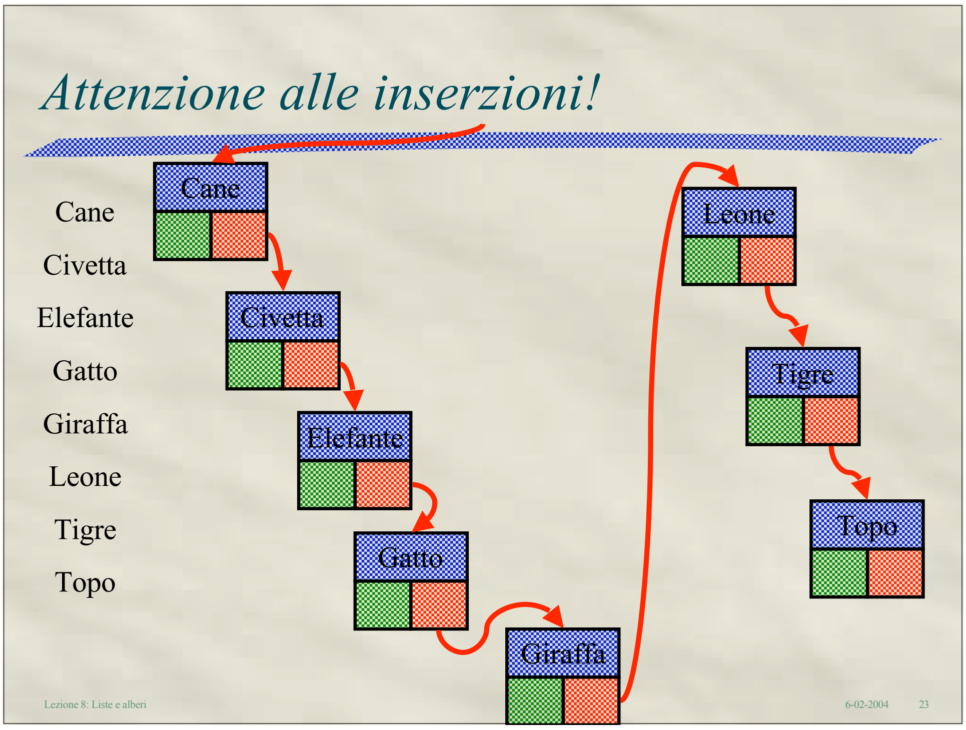
Civetta

Tigre

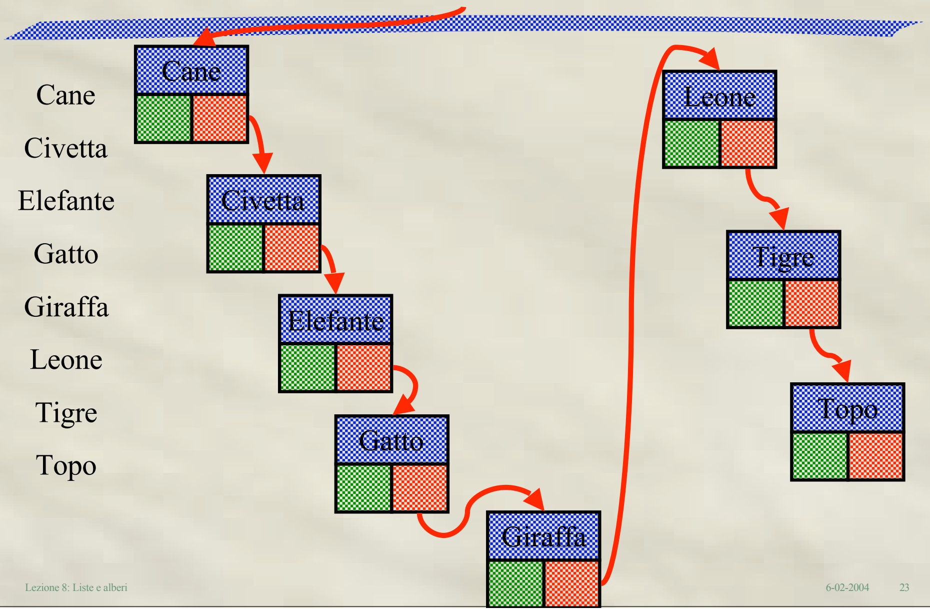
Leone

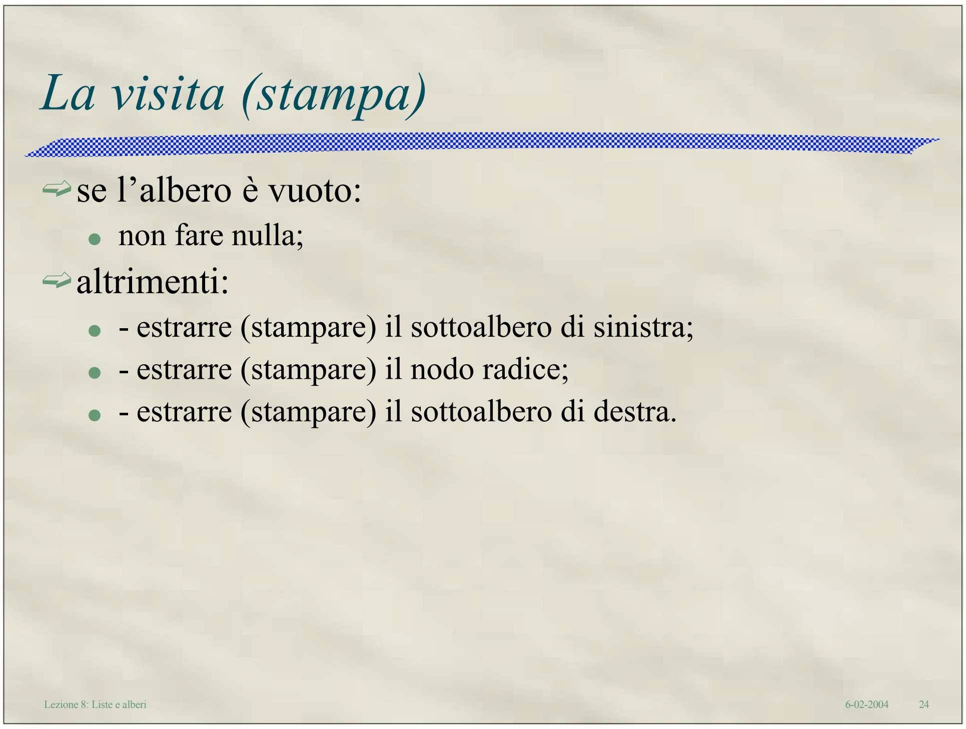
Topo

Giraffa



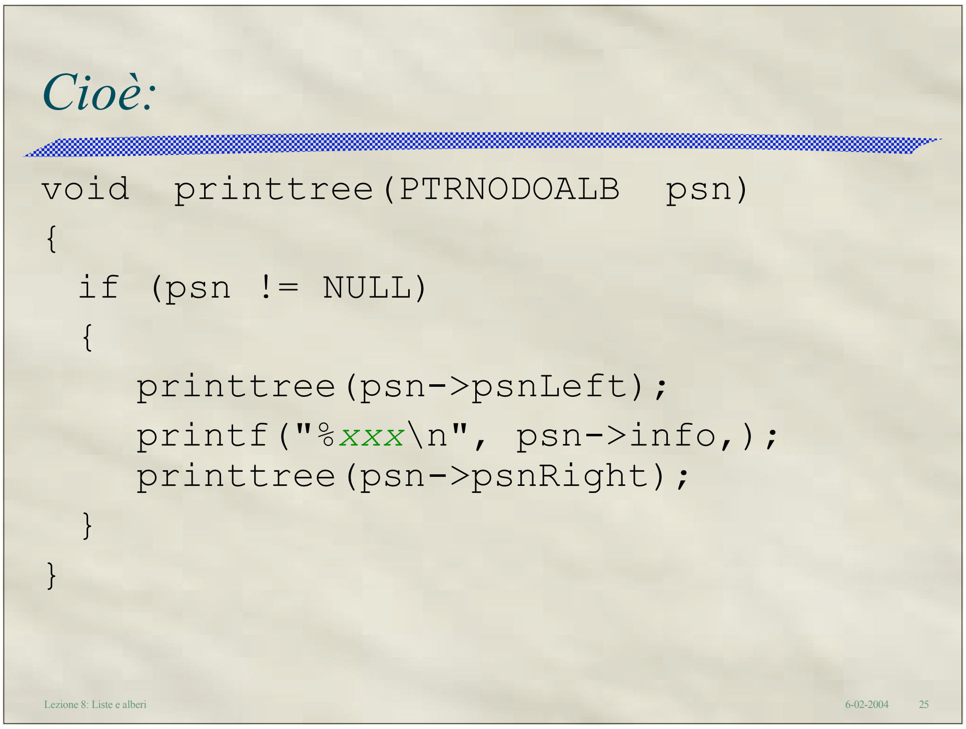
Attenzione alle inserzioni!





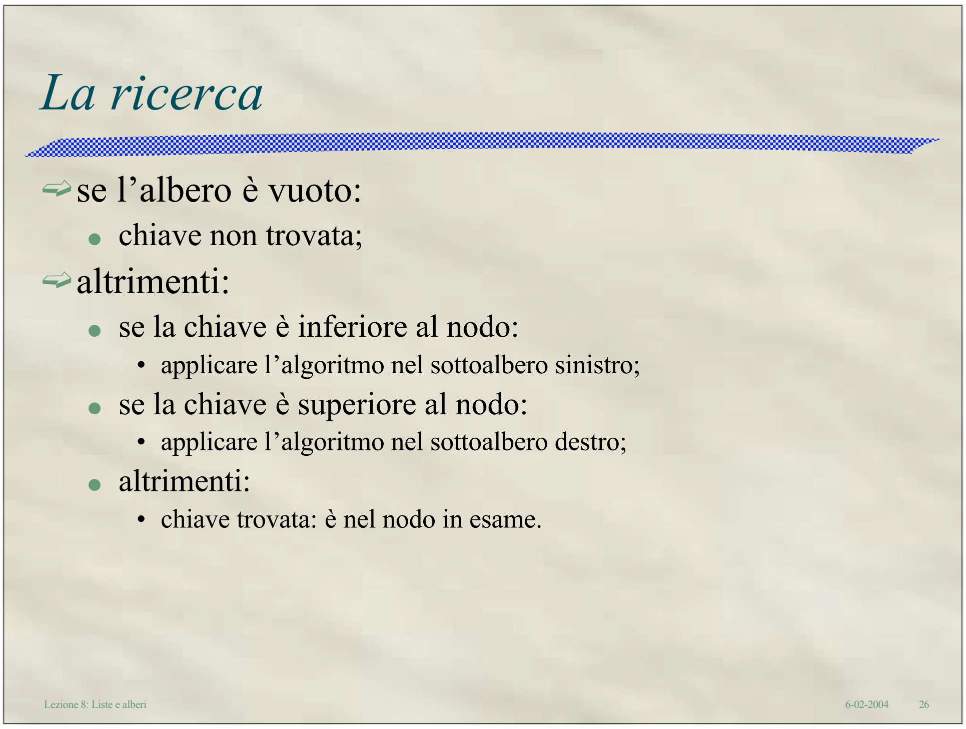
La visita (stampa)

- se l’albero è vuoto:
  - non fare nulla;
- altrimenti:
  - - estrarre (stampare) il sottoalbero di sinistra;
  - - estrarre (stampare) il nodo radice;
  - - estrarre (stampare) il sottoalbero di destra.



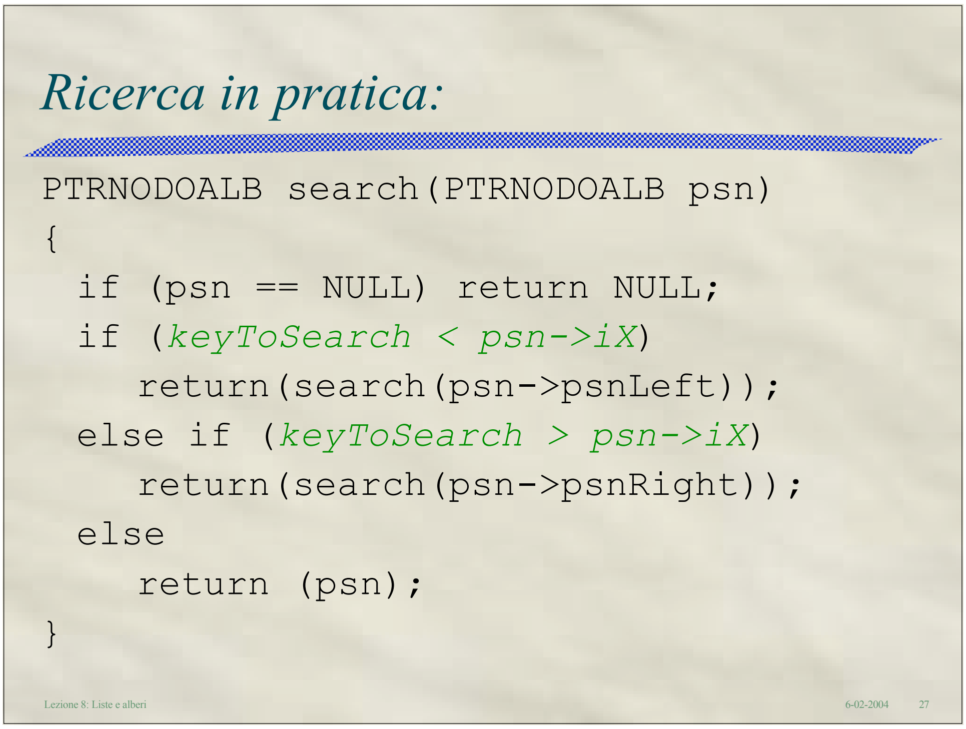
Cioè:

```
void  printtree(PTRNODOALB  psn)
{
  if (psn != NULL)
  {
     printtree(psn->psnLeft);
     printf("%xxx\n", psn->info,);
     printtree(psn->psnRight);
  }
}
```



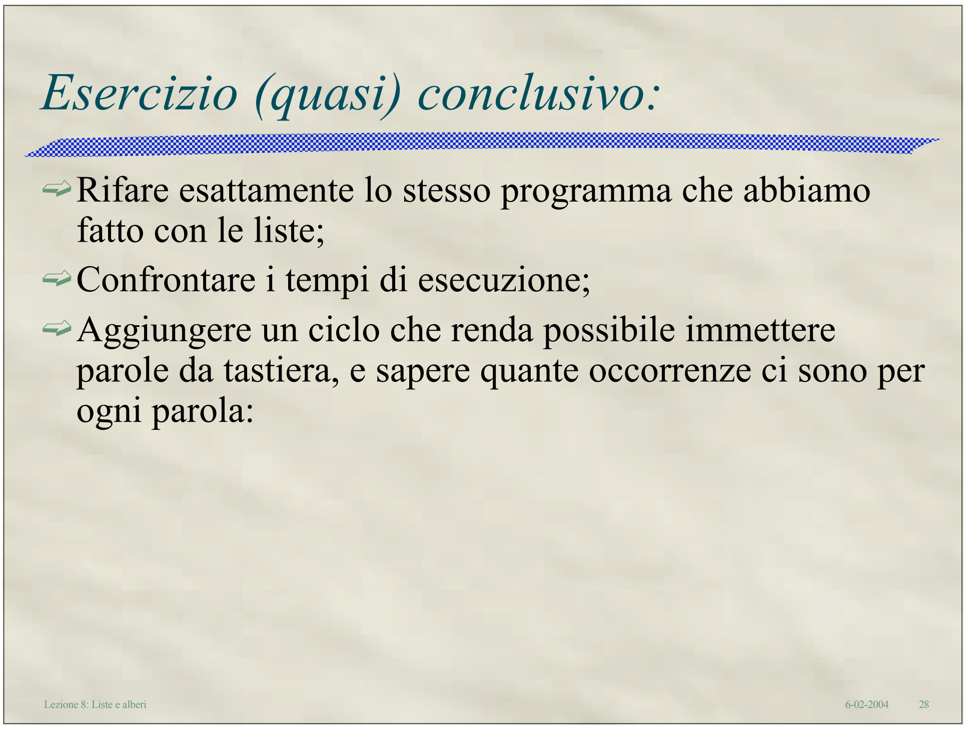
La ricerca

- se l’albero è vuoto:
  - chiave non trovata;
- altrimenti:
  - se la chiave è inferiore al nodo:
    - applicare l’algoritmo nel sottoalbero sinistro;
  - se la chiave è superiore al nodo:
    - applicare l’algoritmo nel sottoalbero destro;
  - altrimenti:
    - chiave trovata: è nel nodo in esame.



Ricerca in pratica:

```
PTRNODOALB search(PTRNODOALB psn)
{
  if (psn == NULL) return NULL;
  if (keyToSearch < psn->iX)
     return(search(psn->psnLeft));
  else if (keyToSearch > psn->iX)
     return(search(psn->psnRight));
  else
     return (psn);
}
```



Esercizio (quasi) conclusivo:

- Rifare esattamente lo stesso programma che abbiamo fatto con le liste;
- Confrontare i tempi di esecuzione;
- Aggiungere un ciclo che renda possibile immettere parole da tastiera, e sapere quante occorrenze ci sono per ogni parola: