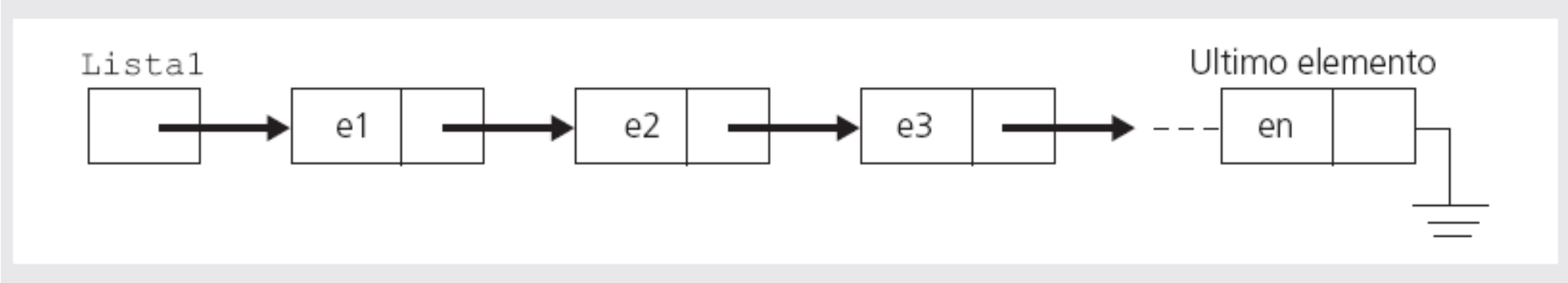
Implementazione di strutture lineari

- La lista, la pila e la coda, così come l'albero e il grafo che vedremo successivamente, possono essere rappresenti nella memoria di un calcolatore utilizzando ***strutture gestite dinamicamente mediante puntatori***.
- L'idea consiste nel concatenare tra loro i vari elementi **`e1, e2, e3, …., en`**, considerando questi ultimi come una struttura costituita da 2 parti (o campi), di cui la prima rappresenta il contenuto informativo dell'elemento e la seconda il puntatore al successivo elemento della struttura.



La seconda parte dell'ultimo elemento (en) ha valore NULL, che significa quindi "fine della lista".

L'inizio della lista è definito da una variabile dello stesso tipo del puntatore al successivo elemento in precedenza citato.

Realizzazione di una lista

Per realizzare la generica struttura ”Lista” sarà sufficiente definire un puntatore ad un generico elemento della struttura:

- Dapprima si dichiara il tipo strutturato `struct EL` utilizzando in C una forma sintattica della clausola struct diversa da quella vista finora (dichiarazione ricorsiva):

```
struct EL {
        TipoInfo     Info;
        struct    EL *Prox;
};
```

- Poi si usa una `typedef` per rinominare il tipo `struct EL` come `ElemLista`:

```
typedef   struct    EL     ElemLista;
```

- Infine, con la terza dichiarazione, si definisce il tipo `Lista Di Elem` come puntatore al tipo `ElemLista`:

```
typedef   ElemLista        *ListaDiElem;
```

Si possono così dichiarare diverse variabili del tipo `ListaDiElem`, che costituiscono altrettanti puntatori a diverse liste (o strutture concatenate):

```
ListaDiElem    Lista1, Lista2, Lista3
```

oppure, abbreviatamene:

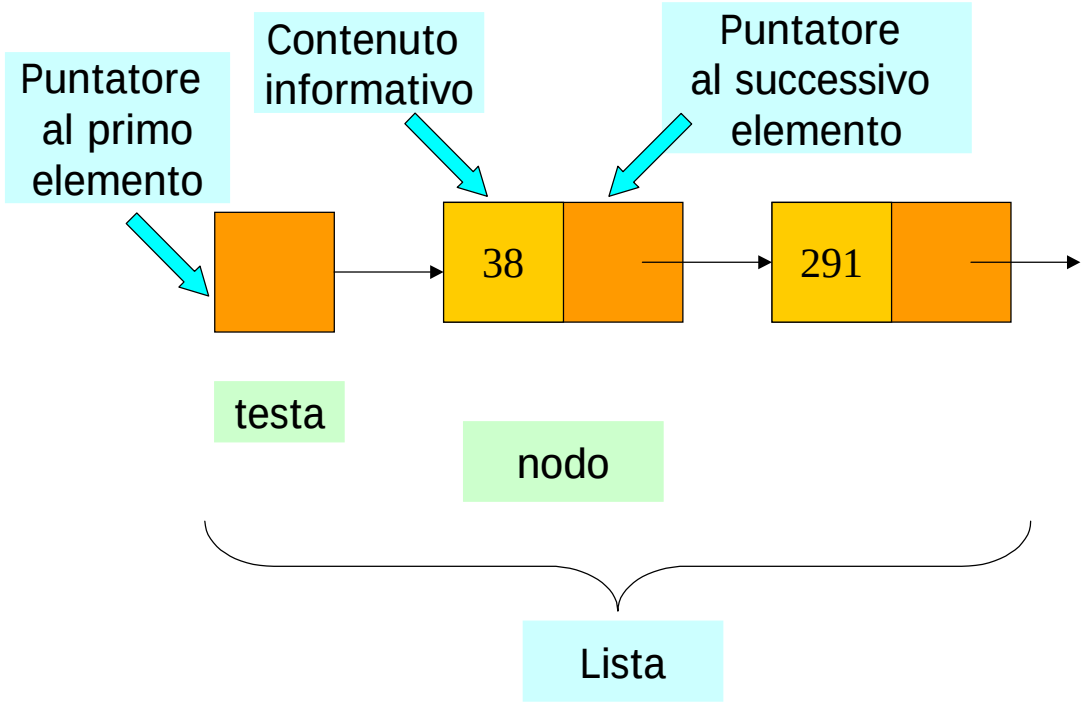
```
ElemLista   *Lista1;
```

o anche:

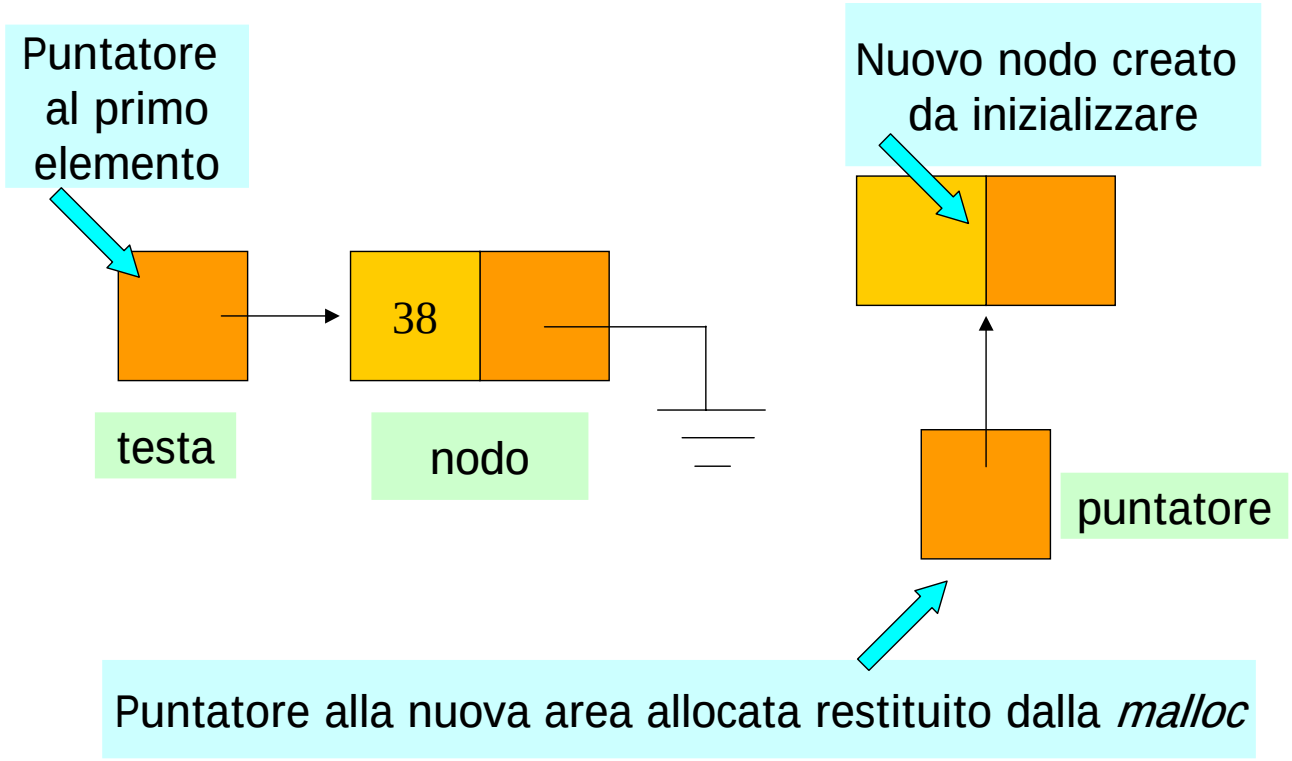
```
struct   EL  *Lista1;
```

Realizzazione di una lista

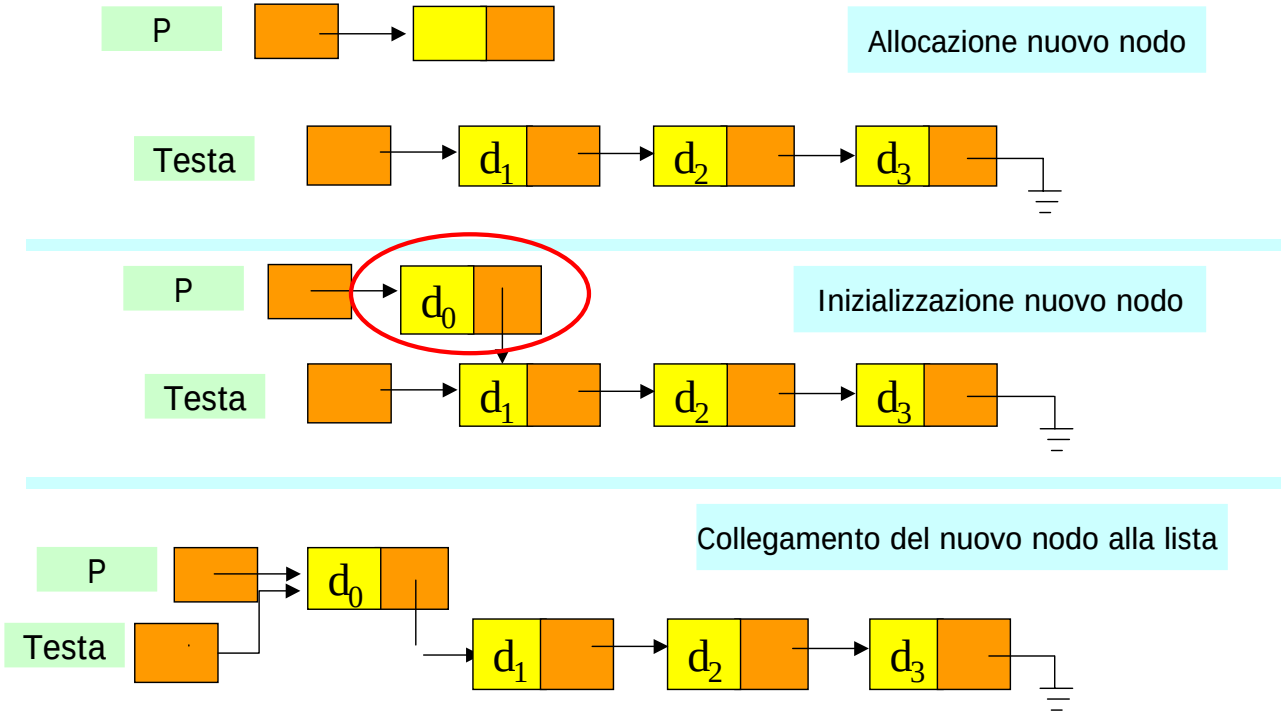
Si rimanda al libro di testo per la realizzazione delle varie operazioni di gestione di una lista, di seguito riportate solo graficamente.



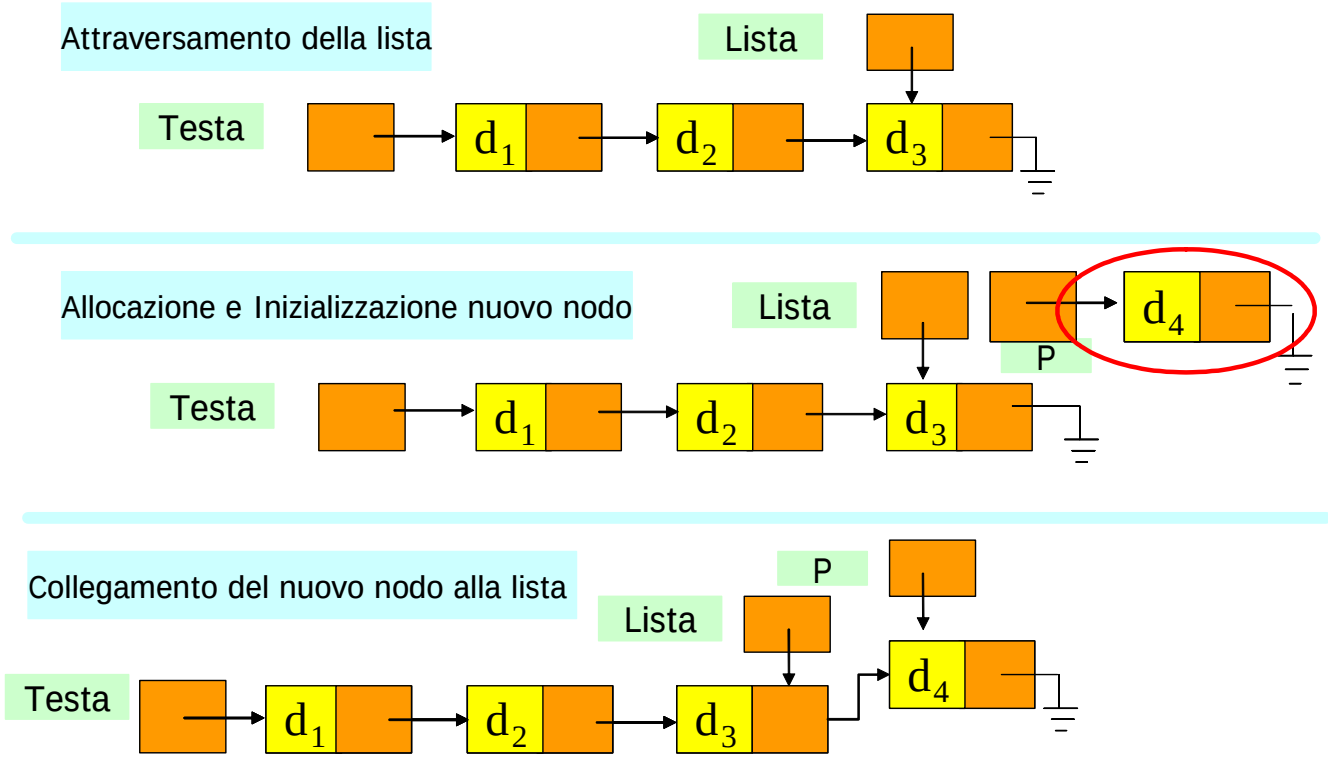
Allocazione della memoria per un nuovo nodo di una lista



Inserimento in testa ad una lista

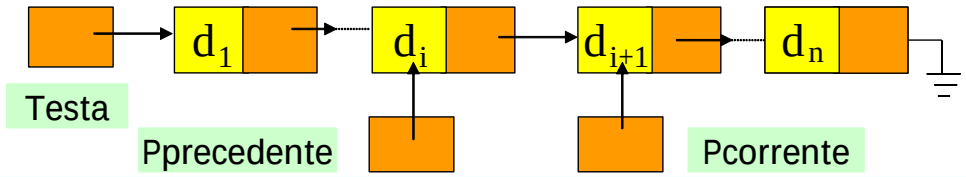


Inserimento in coda ad una lista

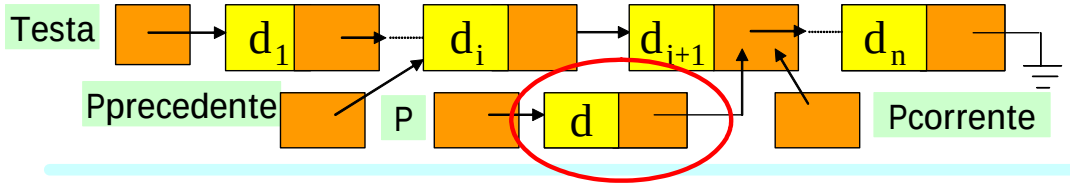


Inserimento in ordine in una lista

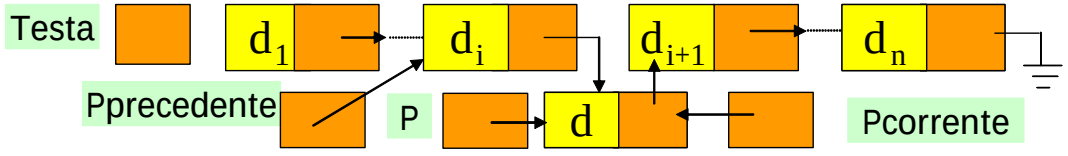
Individuazione della posizione in cui inserire il dato $d_i<d<d_{i+1}$



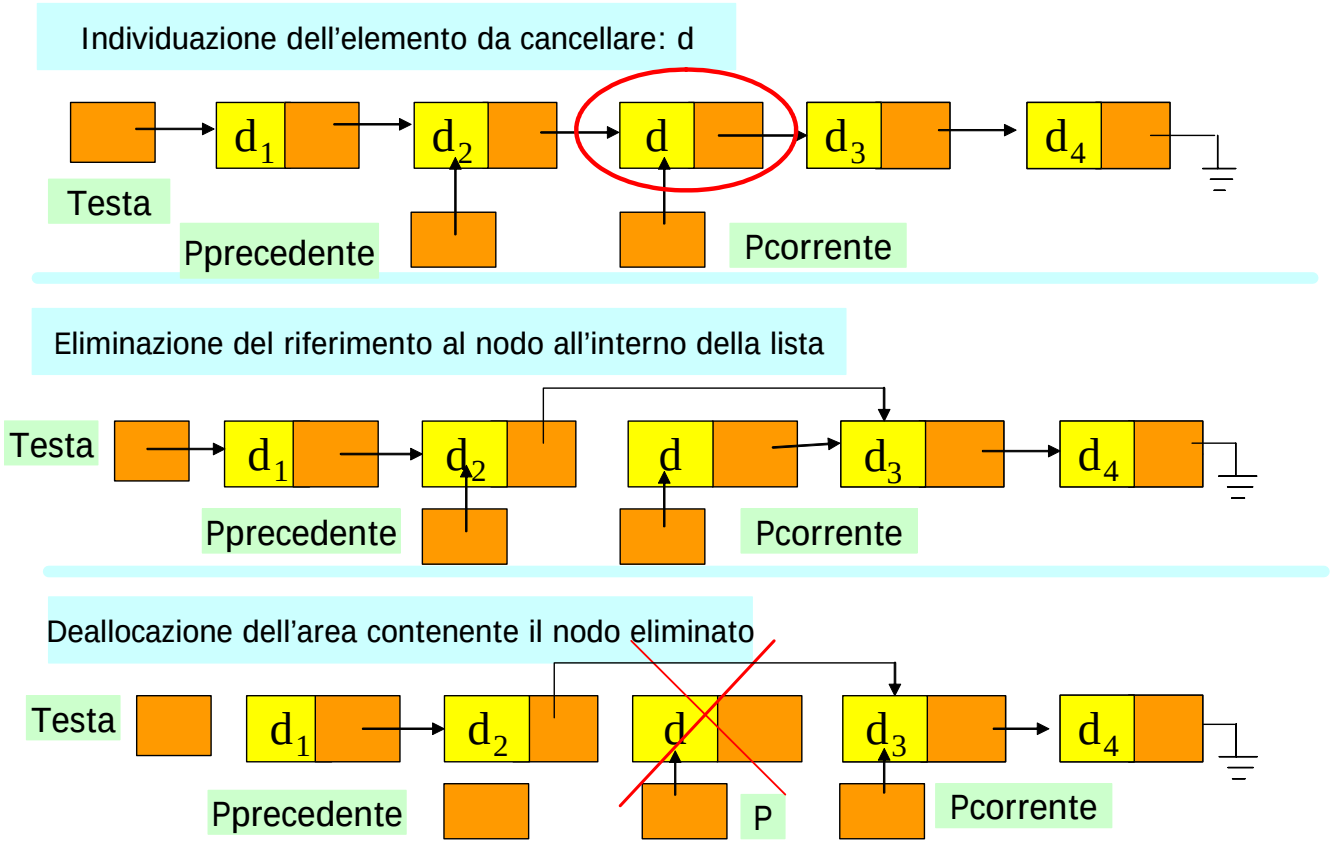
Allocazione e Inizializzazione del nuovo nodo



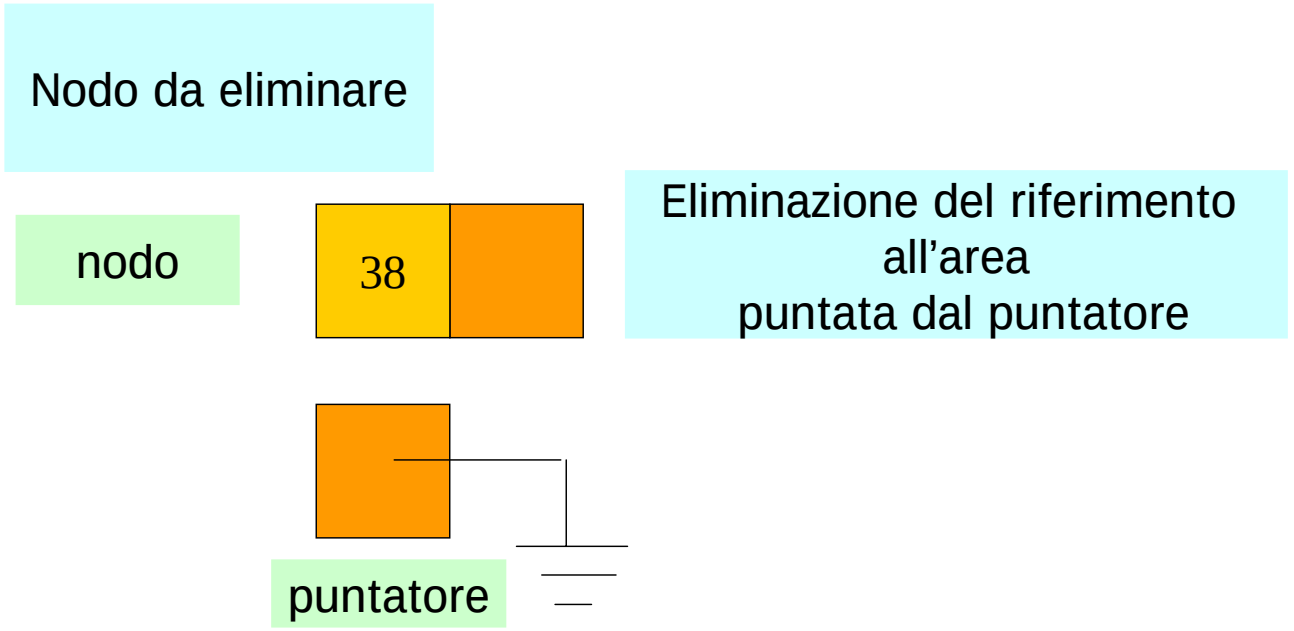
Collegamento del nuovo nodo alla lista



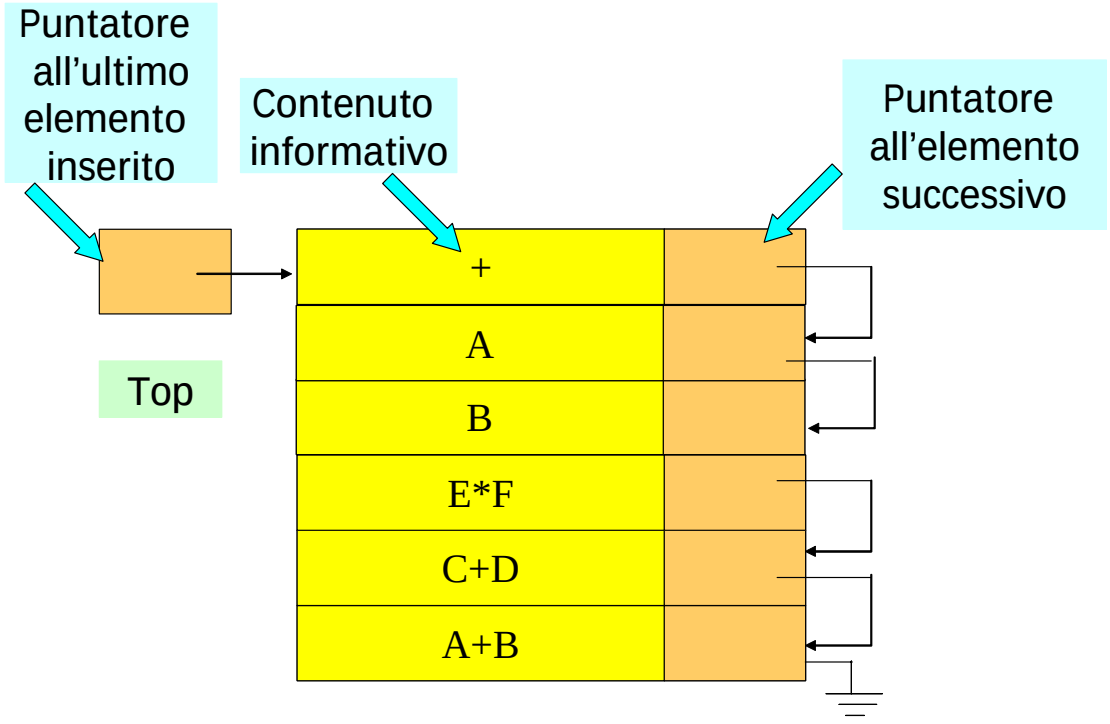
Cancellazione di un nodo in una lista



Rilascio della memoria di un nodo

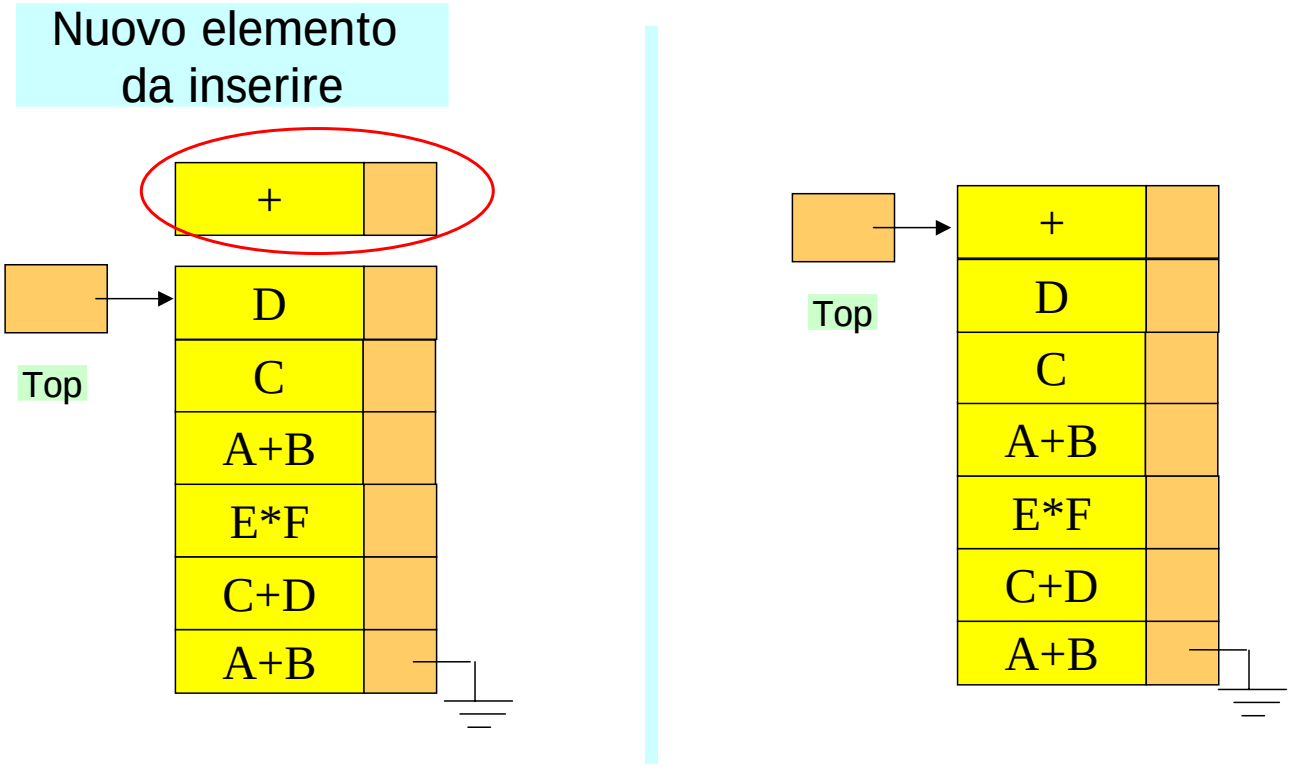


Realizzazione di una pila



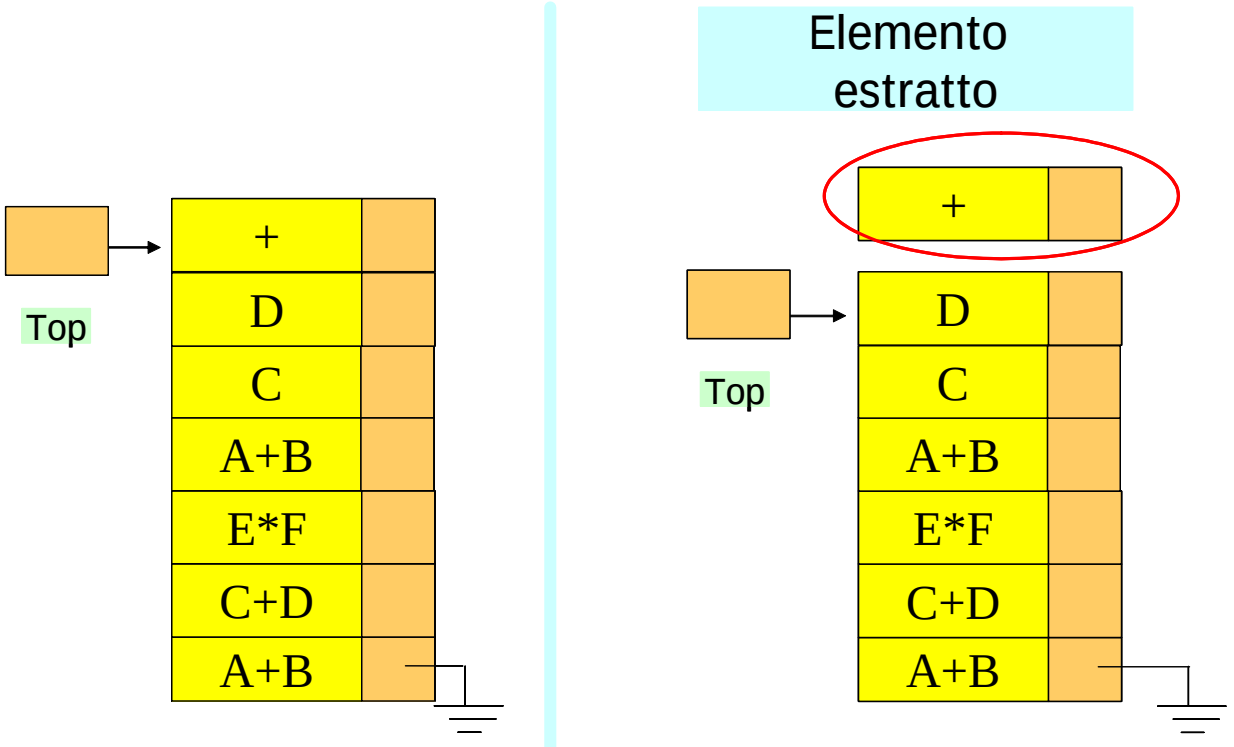
Inserimento di un nuovo elemento in una pila

PUSH

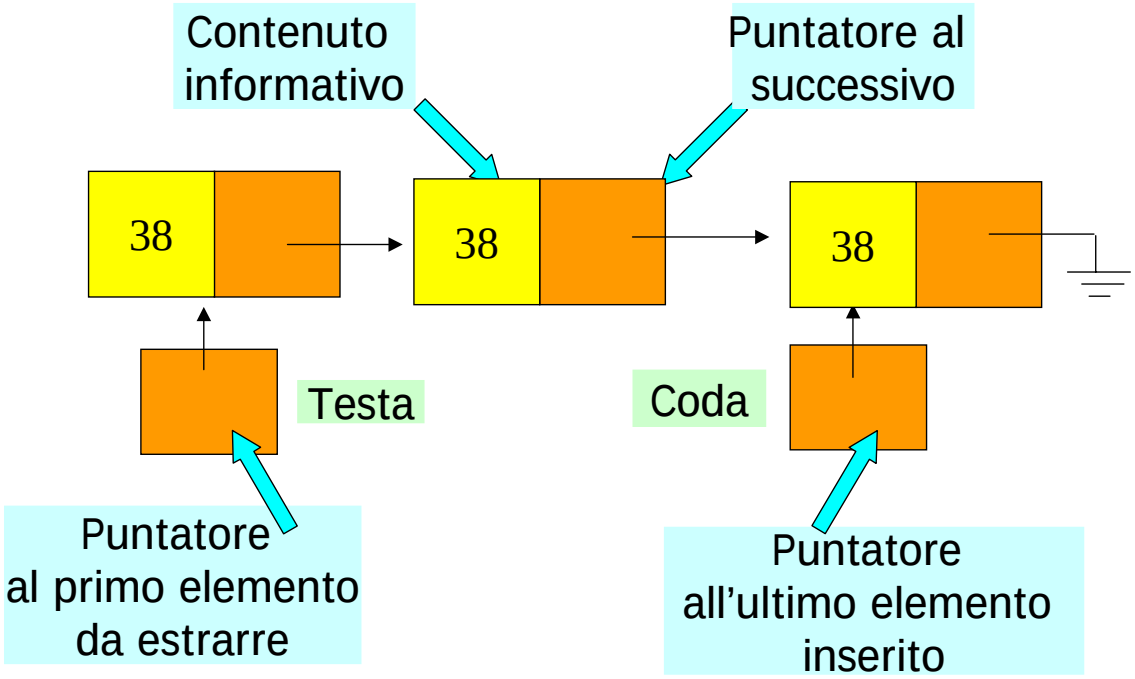


Estrazione di un elemento da una pila

POP



Realizzazione di una coda



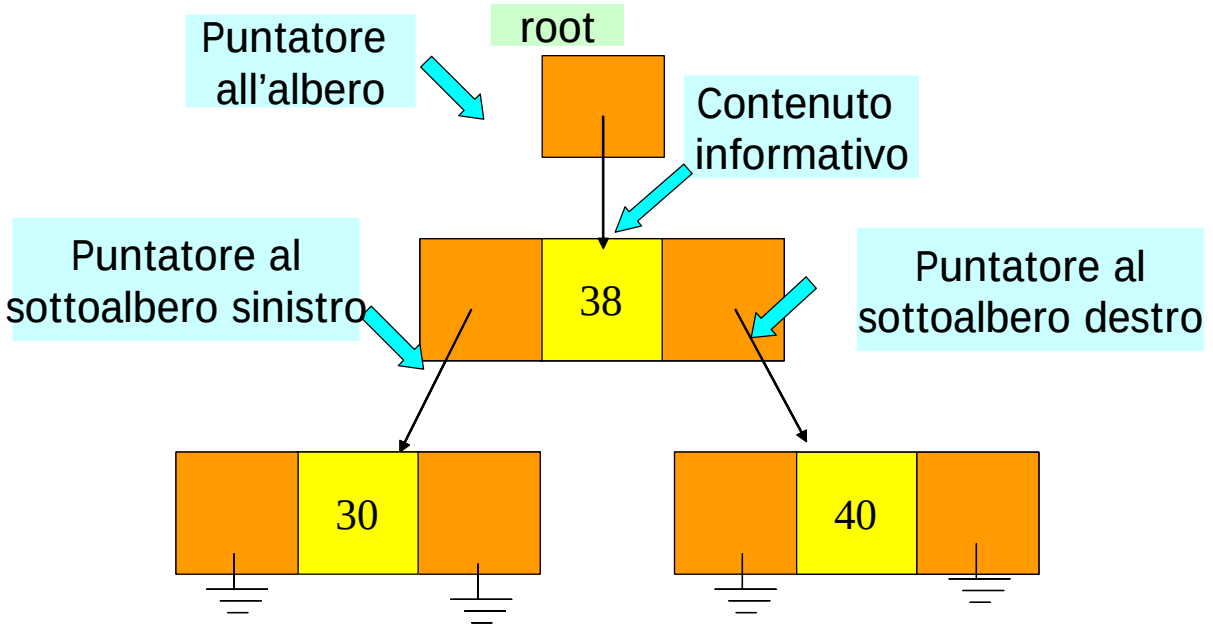
Implementazione di strutture non lineari

Anche strutture non lineari come l'albero possono essere rappresentate ricorrendo all'allocazione dinamica.

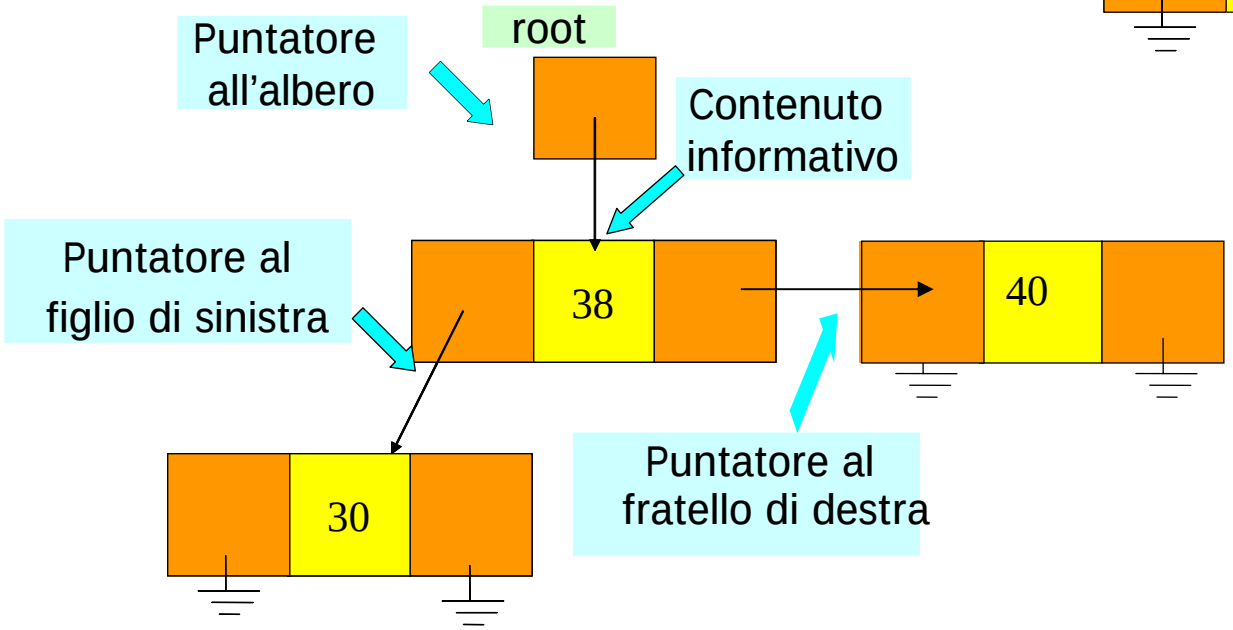
Sarà sufficiente definire una struttura "LISTA" i cui elementi prevedano, oltre al contenuto informativo proprio, ***2 puntatori a elementi successivi***:

- ✓ nel caso di **albero binario**, il primo puntatore indirizzerà al nodo gerarchicamente inferiore di sinistra (o *sottoalbero di sinistra*) e il secondo al nodo gerarchicamente inferiore di destra (o *sottoalbero di destra*);
- ✓ nel caso di **albero n-ario**, il primo puntatore indirizzerà al nodo gerarchicamente inferiore di sinistra (o *figlio di sinistra*) e il secondo al nodo gerarchicamente dello stesso livello di destra (o *fratello di destra*).

Realizzazione di un albero binario

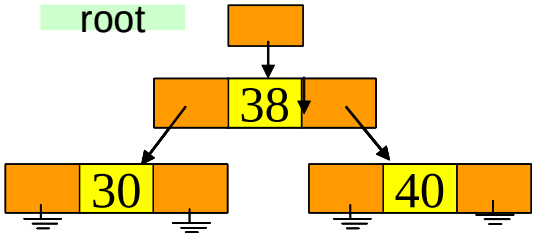


Realizzazione di un albero n-ario

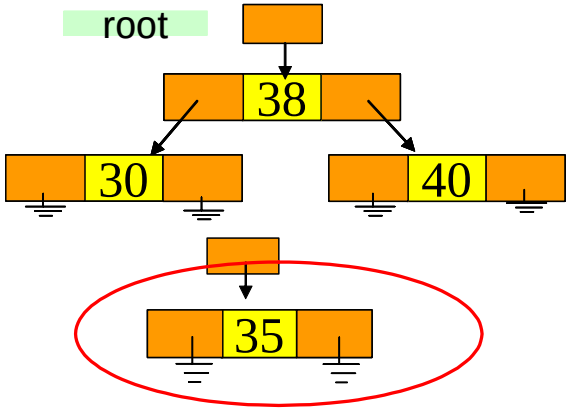


Inserimento di un nuovo nodo in un albero binario

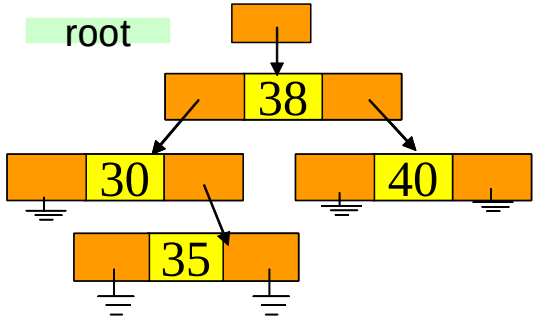
Ricerca del livello in cui inserire il nuovo nodo: 30<**35**<38



Allocazione del nuovo nodo

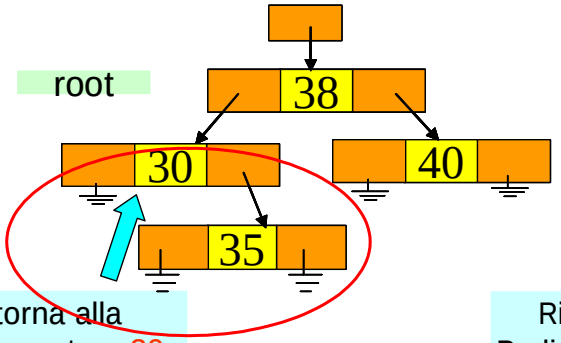


Collegamento del nuovo nodo



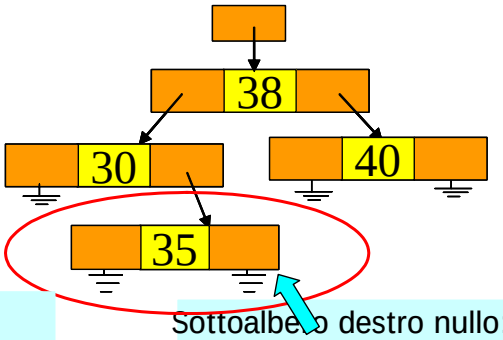
Visita in-ordine di un albero binario

Visita sottoalbero sinistro
fino alle foglie



Ritorna alla
Radice: estrae 30

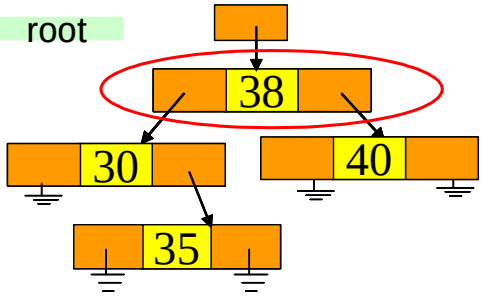
visita sottoalbero destro
della radice 30 fino alle foglie



Ritorna alla
Radice: estrae 35

Sottoalbero destro nullo:
ritorna alla radice 30

Ritorna alla radice: estrae 38



Visita il sottoalbero destro: estrae 40