

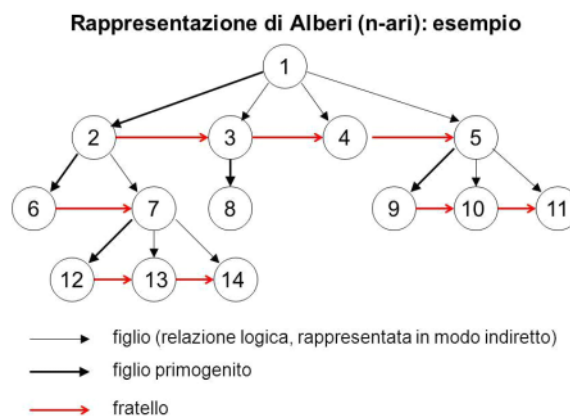
3-Alberi

Struttura dati dinamica non lineare in quanto ogni elemento (o **nodo**) può avere più di un successore (o discendente)

- **Radice**, primo elemento di un albero
- I discendenti diretti di un nodo sono i **figli** del nodo
- Se il nodo n è il figlio del nodo n', allora n' è il **padre** di n
- I nodi con lo stesso padre sono detti **fratelli**
- Un nodo da cui non discende nessun altro nodo si chiama **foglia**
- Ogni nodo che non sia radice è la radice di un albero contenuto nell'albero dato detto **sottoalbero**

Gli **alberi n-ari** possono avere un numero qualsivoglia di figli per ciascun nodo.

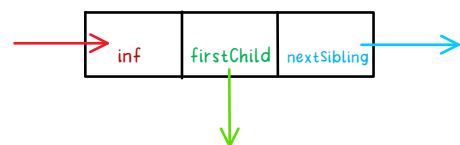
Gli **alberi binari** possono avere 0, 1, o al più 2 figli per ciascun nodo.



30

Ciascun **elemento** (node) contiene:

- un **campo informativo**
- un **puntatore al primo figlio** (firstChild)
- un **puntatore al fratello successivo** (nextSibling)

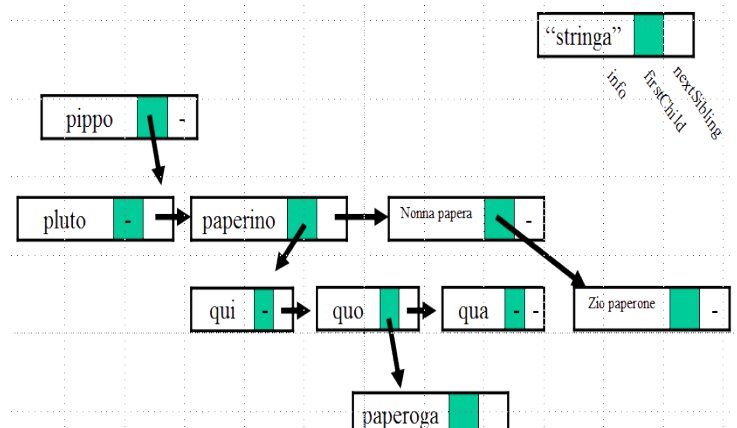


Nodo Albero

```

struct node {
    tipo_inf inf;
    node* parent; //opzionale
    node* firstChild;
    node* nextSibling;
};

typedef node* tree;
  
```



Primitiva NEW_NODE

-Funzione che crea un nuovo nodo con valore informativo i

```
node* new_node (int i) {  
    node* n = new node;  
    n -> firstChild = n -> nextSibling = n -> parent = NULL;  
    return n;  
}
```

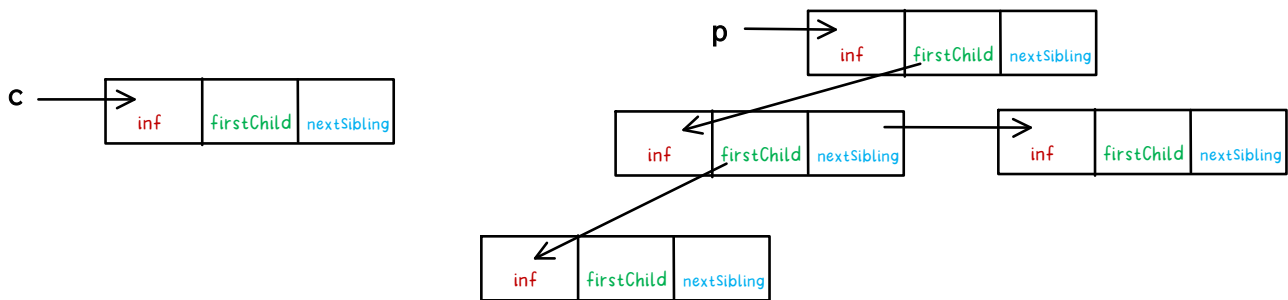
Primitiva INSERT_CHILD

-Funzione che aggiorna p inserendo il sottoalbero radicato in c come primo figlio di p

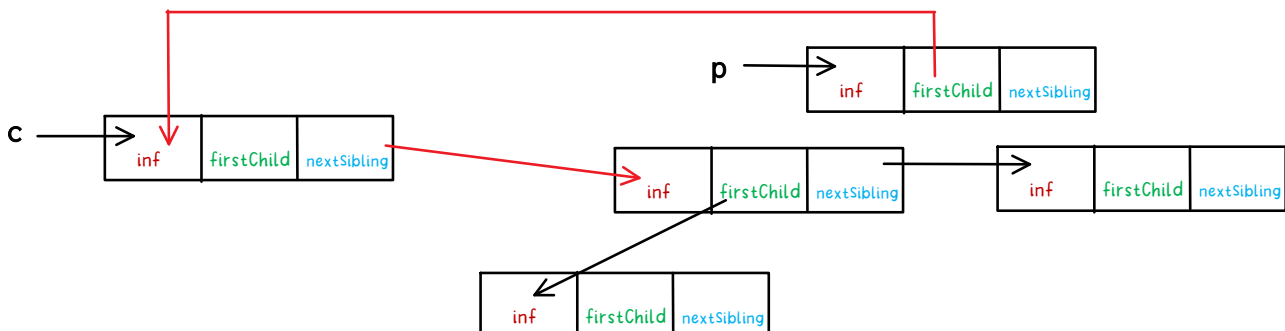
```
void insert_child (tree p, tree c) {  
    c -> nextSibling = p -> firstChild;  
    c -> parent = p; //se il campo esiste  
    p -> firstChild = c;  
}
```

NB. Le istruzioni devono essere eseguite in questo ordine, caso contrario si perderebbe il puntatore ad un eventuale figlio già esistente di p.

Prima dell'inserimento



Dopo l'inserimento

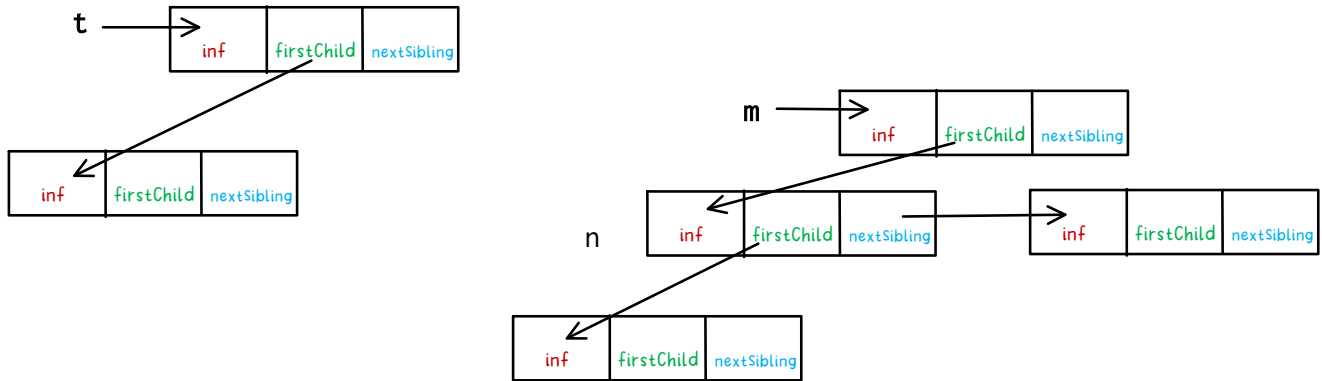


Primitiva NEXT_SIBLING

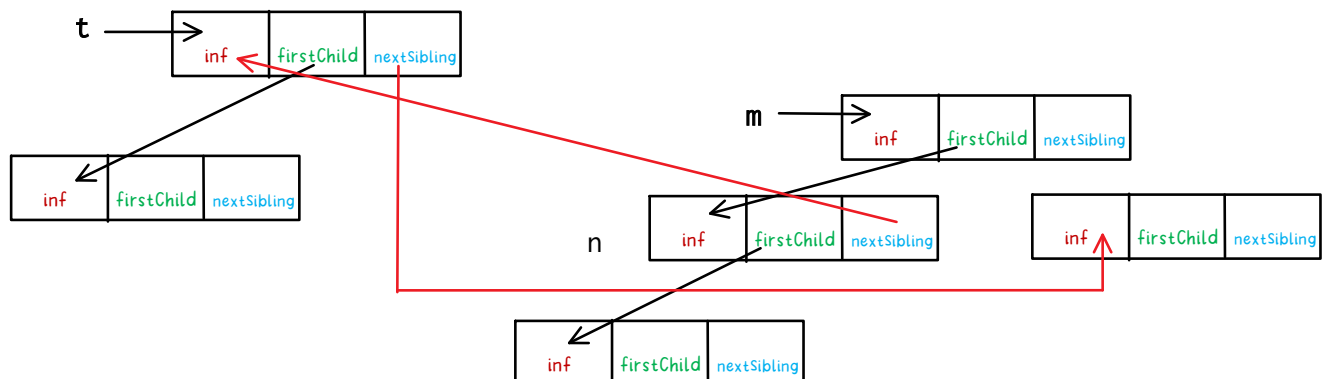
-Funzione che aggiorna n inserendo il sottoalbero radicato in t come fratello successivo di n

```
void insert_sibling (node* n, tree t) {  
    t -> nextSibling = n -> nextSibling;  
    t -> parent = n -> parent; //se il campo esiste  
    n -> nextSibling = t;  
}
```

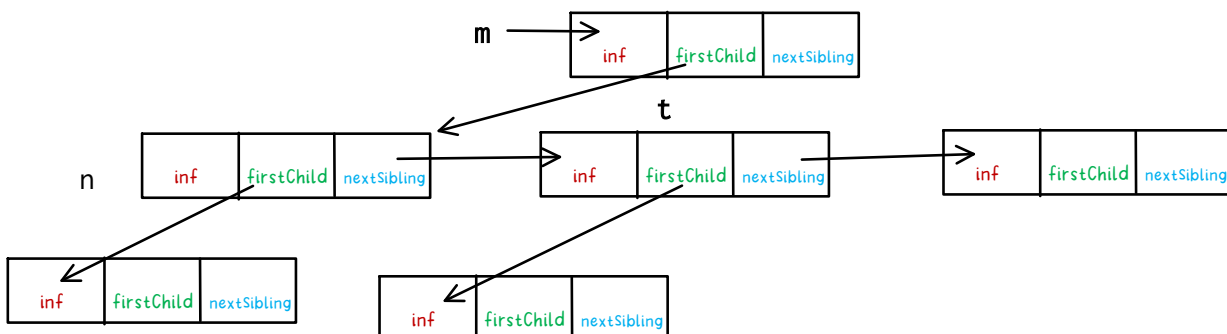
Prima dell'inserimento



L'inserimento



Dopo l'inserimento



Primitiva GET_INFO -Funzione che restituisce il contenuto informativo del nodo n	<pre>int get_info (node* n) { return n -> inf; }</pre>
--	---

Primitiva GET_PARENT -Funzione che restituisce il padre del nodo n	<pre>node* get_parent (node* n) { return n -> parent; }</pre>
--	--

Primitiva GET_FIRSTCHILD -Funzione che restituisce il primo figlio del nodo n, se esiste	<pre>node* get_firstChild (node* n) { return n -> firstChild; }</pre>
--	--

Primitiva GET_NEXTSIBLING -Funzione che restituisce il fratello successive del nodo n, se esiste	<pre>node* get_nextSibling (node* n) { return n -> nextSibling; }</pre>
--	--