

---

---

*Liste*

*Corso di algoritmi*

Giuseppe Lipari

Scuola Superiore Sant'Anna  
Pisa -Italy

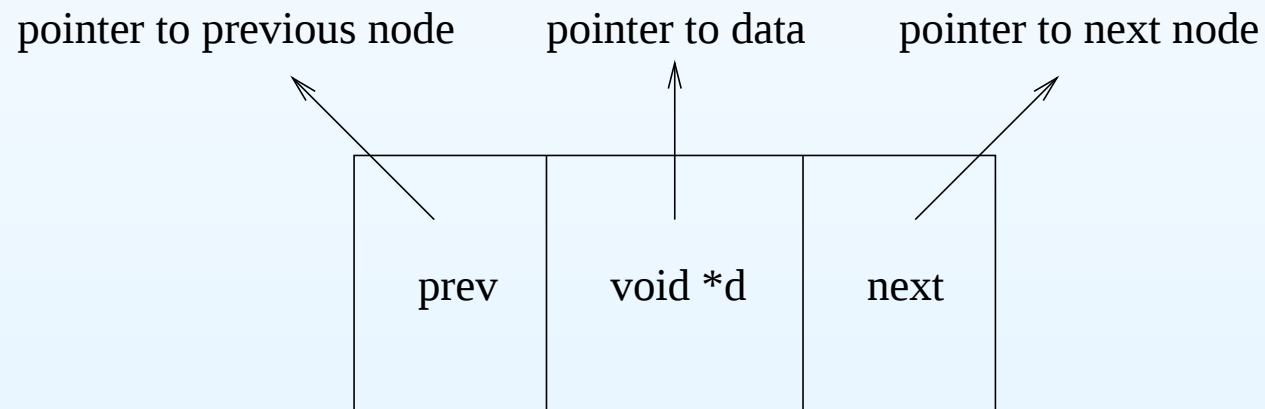
# Liste doppiamente collegate

---

- Si tratta di liste di oggetti lineari.
- Ogni elemento della lista mantiene un puntatore sia all'elemento precedente che all'elemento successivo nella lista.
  - Per questo motivo si parla di liste doppiamente collegate (in inglese: double linked lists).
- Il doppio puntatore serve a facilitare le operazioni di scorrimento degli elementi. Inoltre permette con complessità unitaria sia l'inserimento in testa che in coda alla lista.

# Struttura del nodo

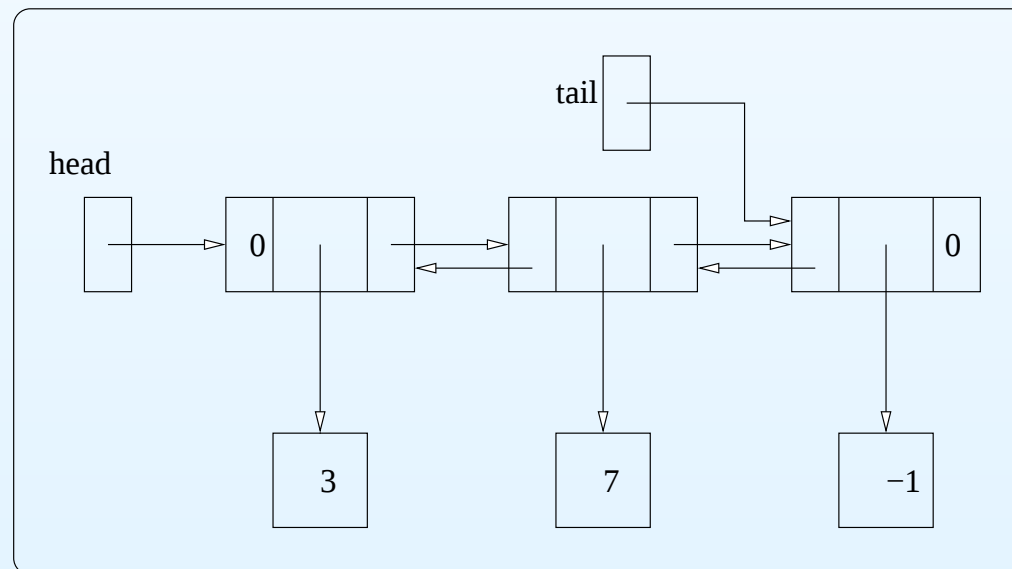
```
struct dll_node {  
    void *d;  
    struct dll_node *next;  
    struct dll_node *prev;  
};  
typedef struct dll_node DLL_NODE;
```



# Esempio di lista

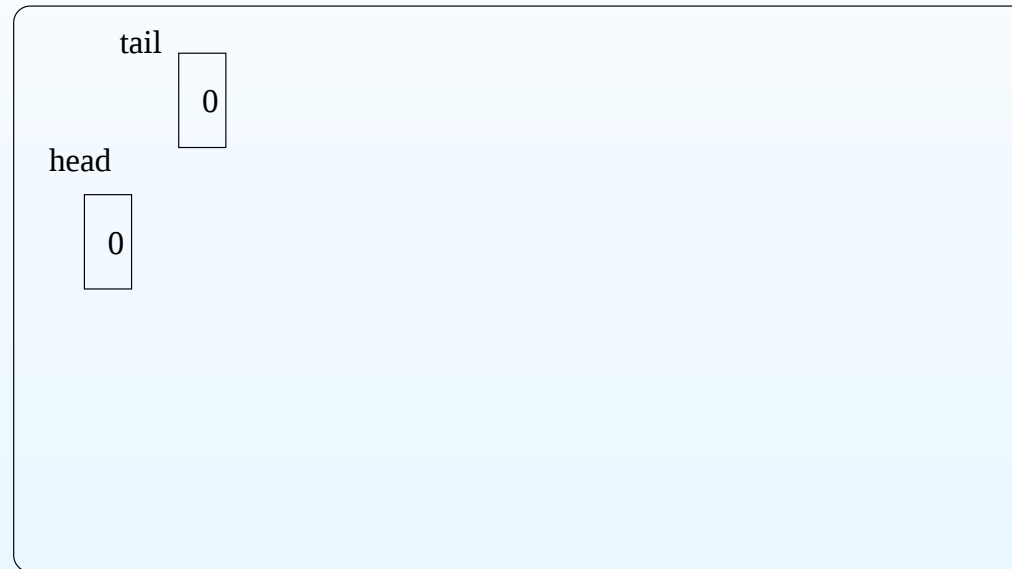
- supponiamo che la lista contenga gli elementi (3, 7, -1);

```
struct dllist {  
    DLL_NODE * head;  
    DLL_NODE * tail;  
};  
typedef struct dllist DLLIST;
```



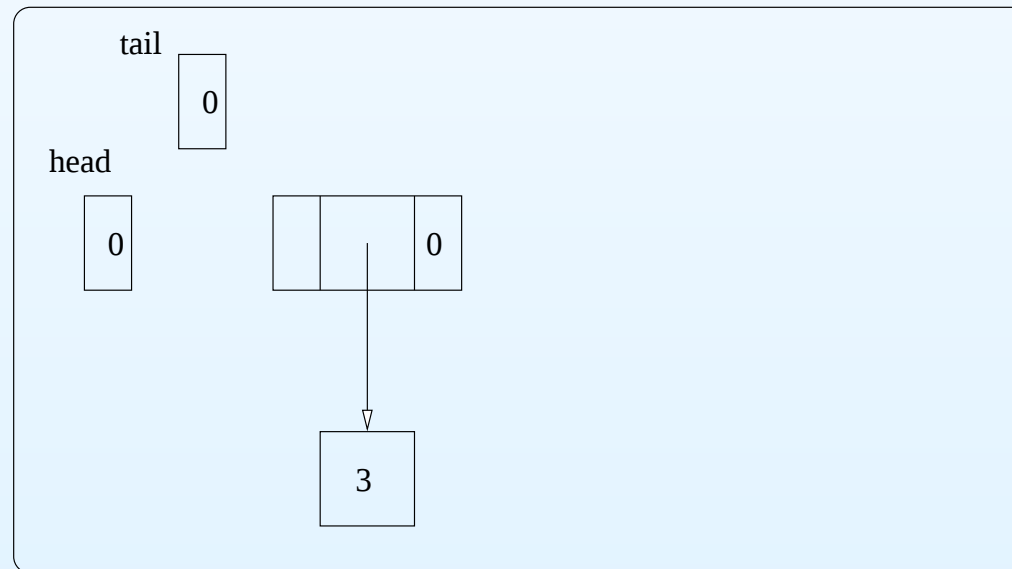
## Inserimento di un elemento

- Inizialmente la lista è vuota. Inseriamo in testa il primo elemento.



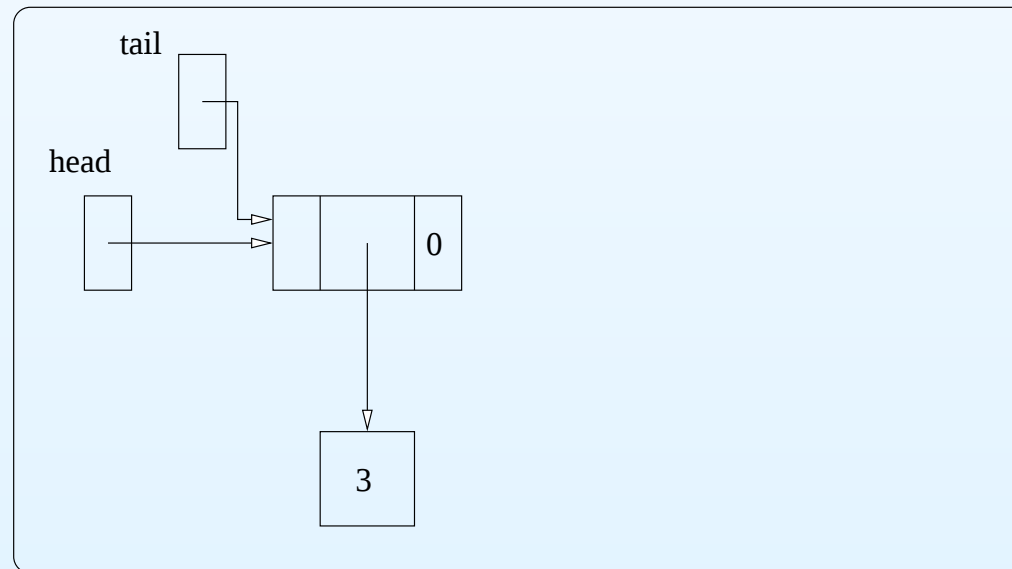
## Inserimento di un elemento - II

```
void dll_insert_head(DLLIST *l, DLL_NODE * p)
{
    p->next = l->head;
    if (l->head != 0) l->head->prev = p;
    else l->tail = p;
    p->prev = 0;
    l->head = p;
}
```



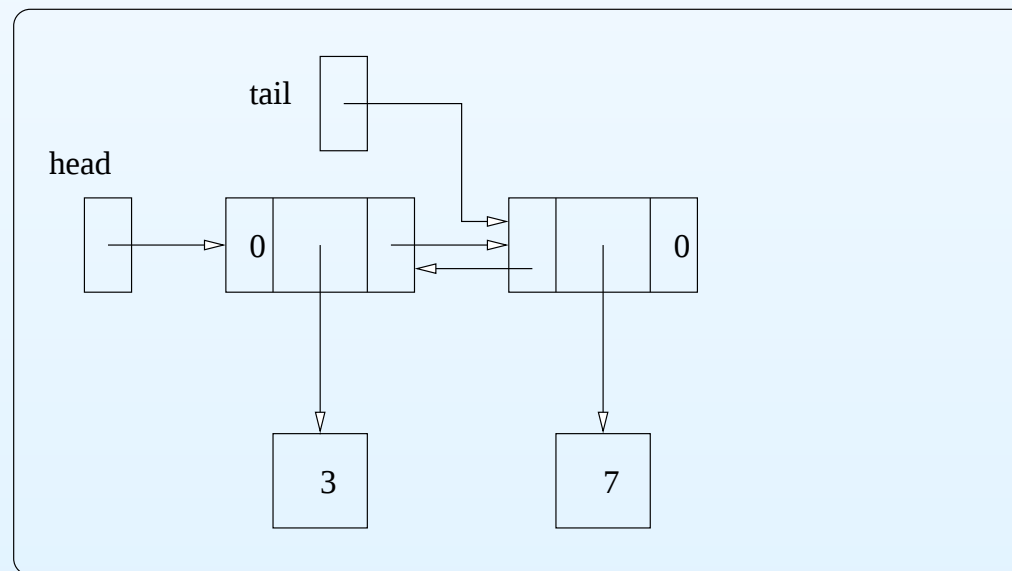
## Inserimento di un elemento - III

```
void dll_insert_head(DLLIST *l, DLL_NODE * p)
{
    p->next = l->head;
    if (l->head != 0) l->head->prev = p;
    else l->tail = p;
    p->prev = 0;
    l->head = p;
}
```



## Inserimento in coda

```
void dll_insert_tail(DLLIST *l, DLL_NODE * p)
{
    p->next = 0;
    p->prev = l->tail;
    if (l->tail != 0) l->tail->next = p;
    else l->head = p;
    l->tail = p;
}
```



## Complessità delle operazioni

---

- Inserimento in testa o in coda:  $O(1)$
- Inserimento in lista ordinata:  $O(n)$
- Ricerca elemento:  $O(n)$

## Indipendenza dal tipo di oggetto

---

- Il nodo in questione contiene come dato un puntatore a un dato di tipo non specificato (void \*).
- Questo metodo ha due vantaggi:
  1. E' indipendente da tipo di dato: possiamo inserire in coda qualunque tipo di dato, da molto semplice a molto complesso;
  2. Poichè il dato esiste già (nella lista inseriamo solo il puntatore), è possibile inserire uno stesso dato in più liste indipendenti.
- Lo svantaggio principale è:
  - se abbiamo un puntatore all'oggetto, è impossibile risalire alla lista che lo contiene.

## Codice sorgente

Nel pacchetto, il codice per implementare una Double Linked List si trova nei file `include/list.h` e `src/list.c`.

Come sempre, nel file header si trova l'*interfaccia* del modulo, e va incluso da qualunque altro modulo voglia utilizzare delle dll.

Ecco le funzioni offerte dal modulo (la documentazione si trova in `code/docs/html/index.html`).

```
DLL_NODE * dll_create_node(void *data);
DLLIST * dll_create_list(void);
void dll_insert_head(DLLIST *l, DLL_NODE * pnode);
void dll_insert_tail(DLLIST *l, DLL_NODE * pnode);
DLL_NODE * dll_extract_head(DLLIST *l);
DLL_NODE * dll_extract_tail(DLLIST *l);
DLL_NODE * dll_find_elem(DLLIST *l, void *d, PCOMPFUNC pf);
void dll_free_node_all(DLL_NODE * p);
void dll_free_node(DLL_NODE * p);
void dll_free_list_all(DLLIST * p);
void dll_free_list(DLLIST * p);
```