Liste concatenate semplici

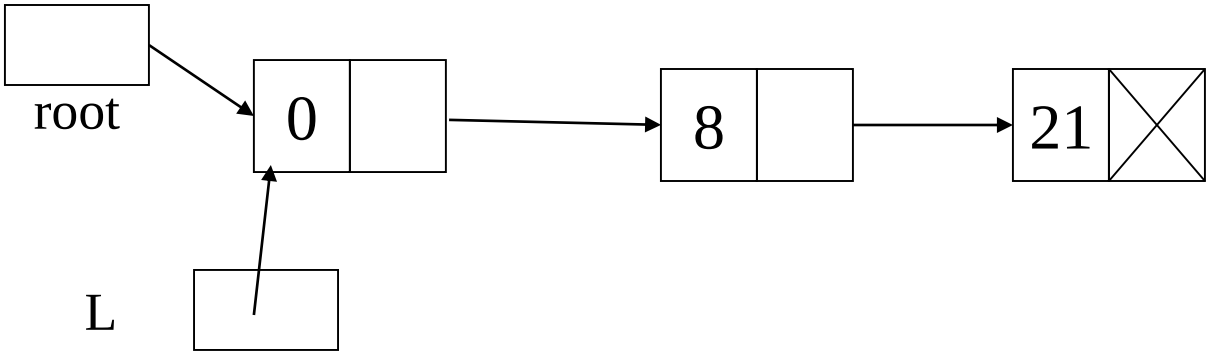
Obiettivi:

- Presentare la realizzazione collegata (puntatori a strutture) di liste semplici

Creazione di una lista

Il ***main*** da realizzare deve leggere la sequenza di interi e inserire ogni elemento letto in una ***lista***

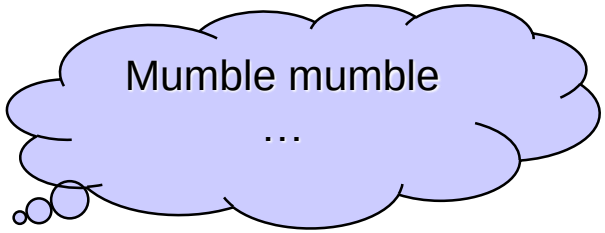
Per ogni inserimento, **`malloc`** e aggiustamento dei puntatori



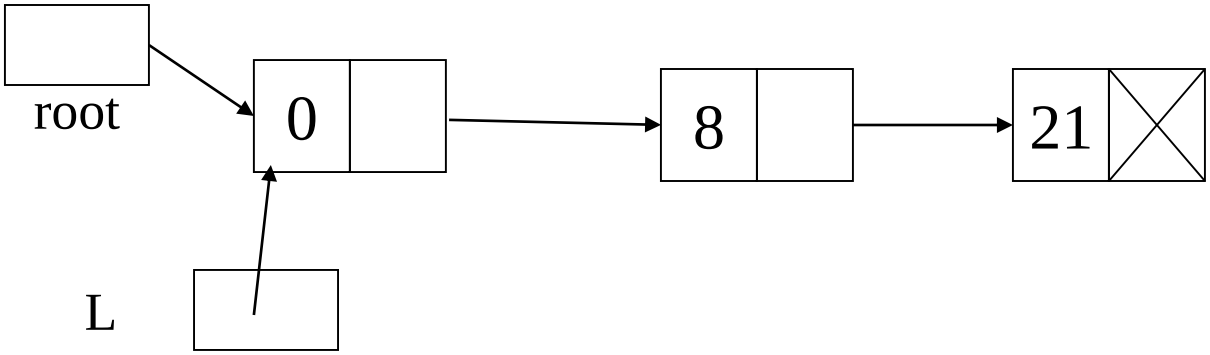
root inizialmente NULL

Stampa a video di una lista

Il ***main*** da realizzare deve leggere la sequenza di interi e inserire ogni elemento letto in una ***lista***



Poi stampiamo la sequenza …
(facciamo insieme anche questo secondo passo)



Non c'è un "indice" da incrementare come per vettori

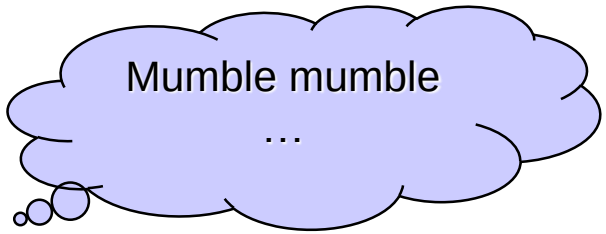
Rivediamo il programma

Abbiamo visto queste due funzioni per inserire un elemento i in testa ad una lista L:

```
root = cons( i, root);
```

e per stampare il contenuto di una lista L:

```
showList(root);
```

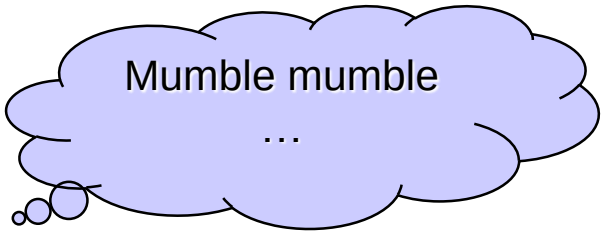


Riscriviamo il ***main*** utilizzando chiamate a ***cons*** e ***showList***

Cerchiamo nella lista

Aggiungiamo al ***main*** che utilizza già chiamate a ***cons*** e ***showList*** anche

la lettura da input di un valore intero



che viene poi cercato in lista con la funzione ***member***

e visualizziamo a video se trovato o meno
if (member(i,root)) printf(“%s”, “trovato”);

ESERCIZIO: ricerca in una lista

```
int member(int e, list l) {
        int trovato = 0;
        while ((l!=NULL) && !trovato)
                if (l->value == e) trovato = 1;
                else l = l->next;
        return trovato;
}
```

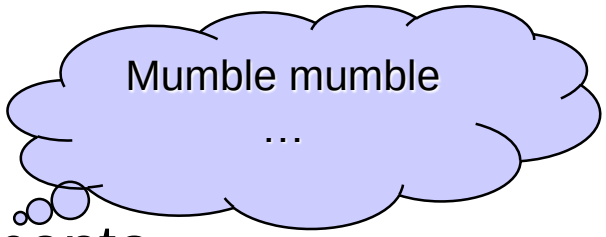
È una ***scansione sequenziale,*** **quale complessità?**

- nel caso ***peggiore***, occorre scandire l'intera lista *O(N)*
- nel caso ***migliore***, è il primo elemento *costante*
- nel caso ***medio***, proporzionale a N/2 *O(N)*

Esercizio proposto: progettare e implementare una soluzione ricorsiva (in Laboratorio – **Esercitazione**)

Creazione di una lista inserendo in fondo

Aggiungiamo al ***main*** che utilizza già chiamate a ***cons*** e ***showList*** anche ***cons_tail***



Lettura di una sequenza da input e costruzione di due liste: root con inserimento in testa e L2 in fondo

Stampa di root e L2 con ***showlist***

Come si presentano le due sequenze visualizzate?

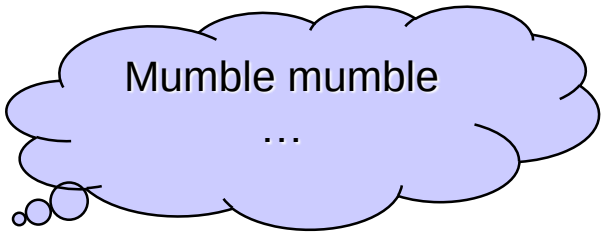
Creazione di tre liste

Riscriviamo il ***main*** creando tre liste, L1, L2, e L3, usando ***cons***, ***cons_tail*** e ***insord_p*** per inserire l'elemento letto:

```
L1= cons( i, L1);
L2= cons_tail( i, L2);
L3= insord_p( i, L3);
```

e stampiamole poi con tre chiamate a:

```
showList( );
```



Leggiamo un intero e cerchiamolo nelle tre liste:

```
member(i, L );
```

Per la lista ordinata L3, cambia la complessità della **`member`**?

ESERCIZIO : ricerca in una lista

```
int member(int e, list l) {
        int trovato = 0;
        while ((l!=NULL) && !trovato)
                if (l->value == e) trovato = 1;
                else l = l->next;
        return trovato;
}
```

È sempre una ***scansione sequenziale,*** **anche se la lista è ordinata !**

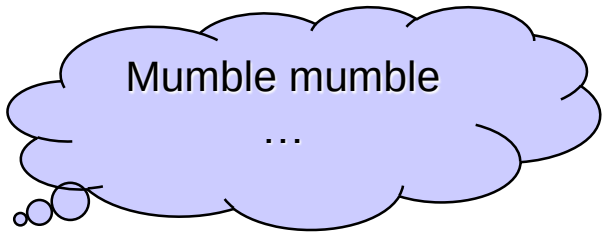
- nel caso ***peggiore***, occorre scandire l'intera lista *O(N)*
- nel caso ***medio***, proporzionale a N/2 *O(N)*

Nota Bene: L'accesso agli elementi della lista è sempre sequenziale (dal primo all'ultimo), non c'è accesso diretto a elementi intermedi!

DO IT!

Riscriviamo il ***main*** creando tre liste, L1, L2, e L3, usando ***cons***, ***cons_tail*** e ***insord (insord_p)*** per inserire l'elemento letto:

```
L1= cons( i, L1);
L2= cons_tail( i, L2);
L3= insord( i, L3);
```



e stampiamole poi con tre chiamate a:

```
showList( );
```

Come si presentano le tre sequenze visualizzate?

Esercizi per Laboratorio

- Scrivere una versione ricorsiva della funzione **`showList`**
- Scrivere una versione ricorsiva della funzione **`member`**

- Scrivere una versione iterativa e una ricorsiva della funzione **`lenght`** che calcoli la lunghezza di una lista
- Scrivere una versione iterativa e una ricorsiva della funzione **`sumlist`** che calcoli la somma degli elementi di una lista di interi
- Definire una funzione **`subList`** che, dato un intero positivo n e una lista l, restituisca una lista che rappresenti *la sotto-lista di quella data a partire dall'elemento n-esimo*

ESEMPIO: l = [1, 13, 7, 9, 10, 1]
subList(2, l) = [7, 9, 10, 1]