Parte 4D

Liste doppie

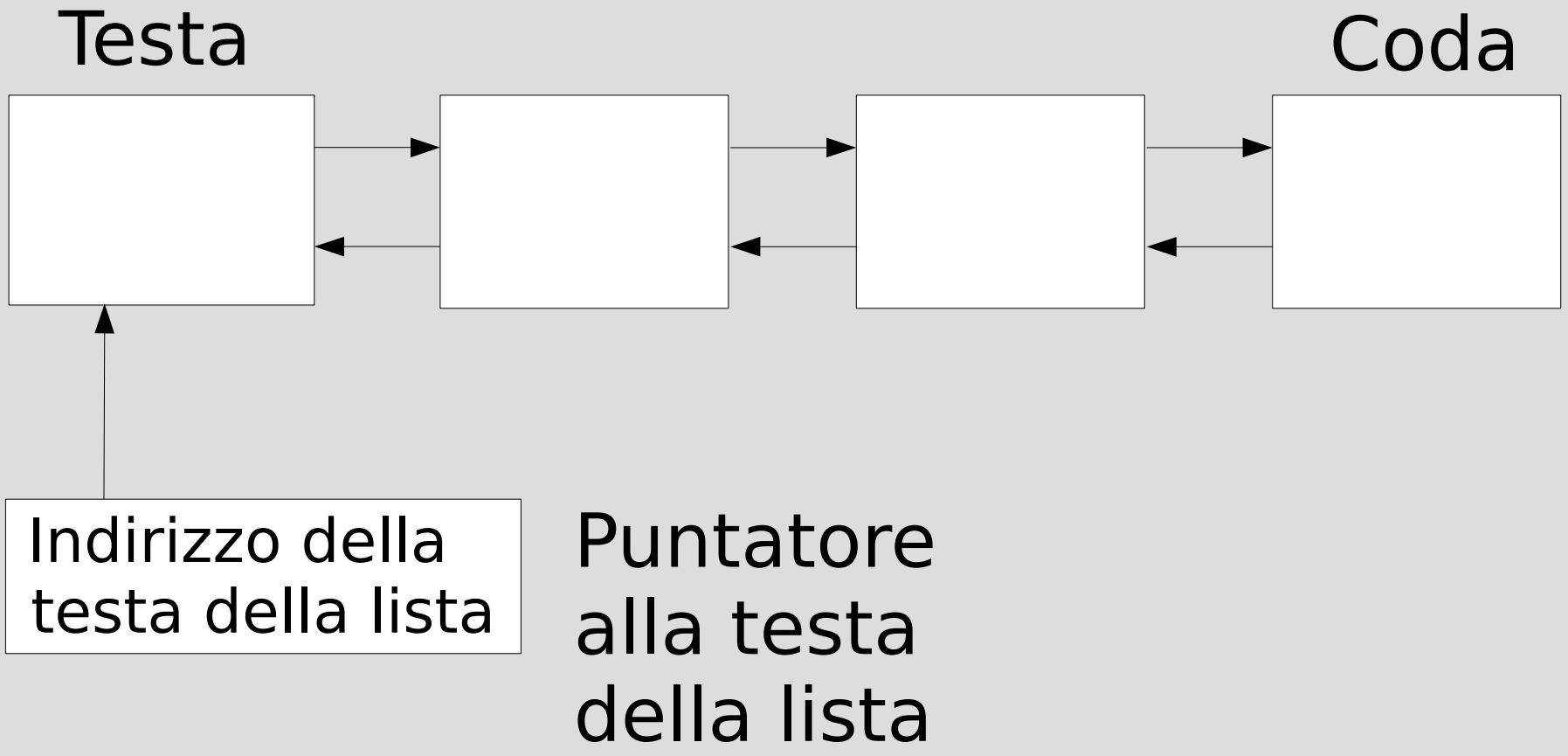
R. Magritte – Transfer, 1966

Liste doppie

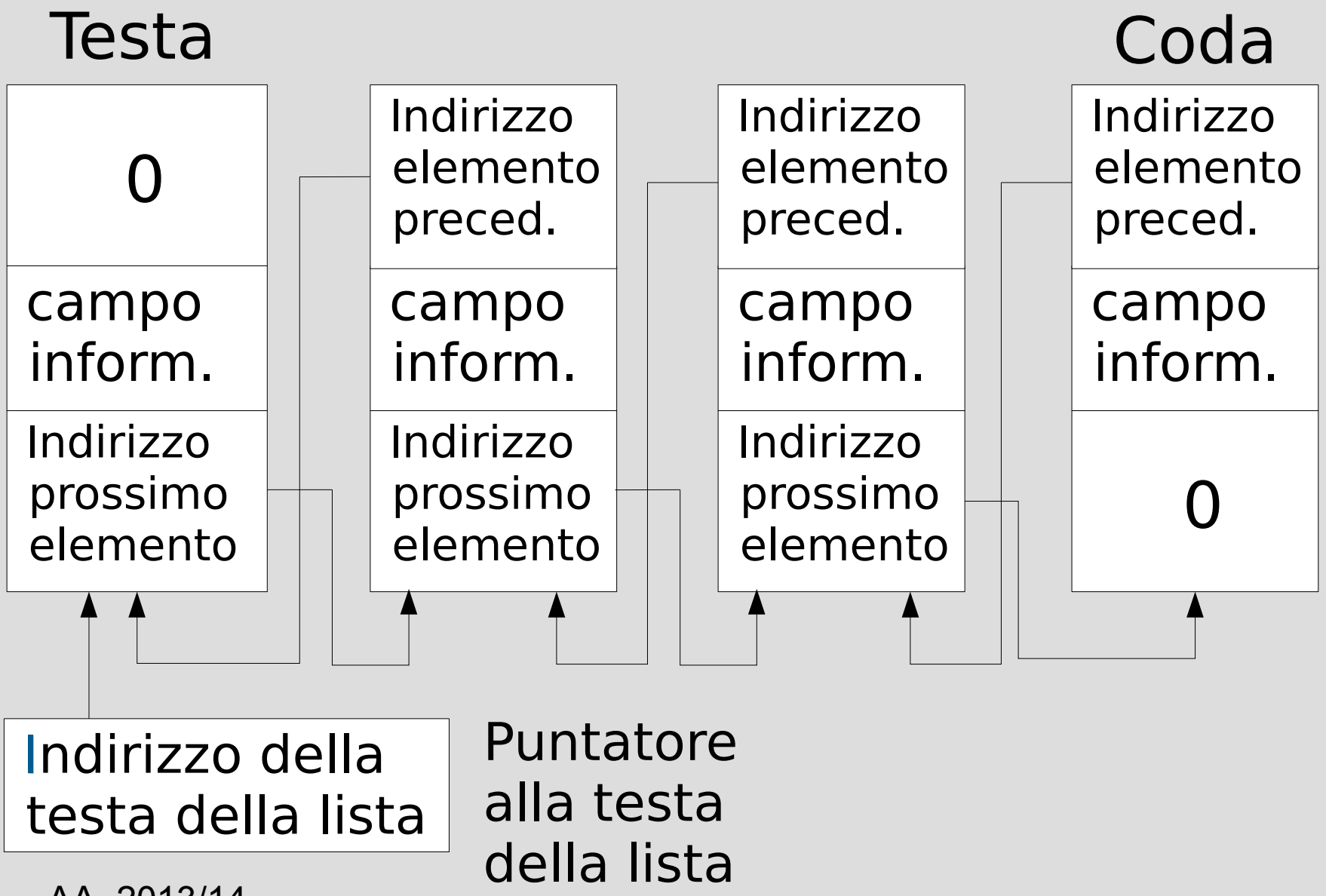
- Lista ***singolarmente concatenata*** o ***lineare***: ciascun elemento contiene solo un puntatore al prossimo elemento
- Lista ***doppiamente concatenata*** o ***doppia***: ciascun elemento contiene due puntatori
  - un puntatore al prossimo elemento
  - un puntatore all'elemento precedente

Schema lista doppia

- Ciascun elemento punta sia al precedente che al successivo



Schema lista doppia



Struttura dati

```
struct elem {
int inf;
elem* pross; // punt. al prossimo elem
elem* prec; // punt. al precedente elem
} ;

elem *testa; // puntatore alla testa

// oppure:
typedef elem* lista;
lista testa;
```

Stampa della lista

```
void stampalista(lista p)
{
while (p != 0) {
cout<<p->inf<<" " ; // stampa valore
p = p->pross;     // spostamento sul
                  // prossimo elemento
}
cout<<endl;
}
```

Identica alla stampalista per liste semplici

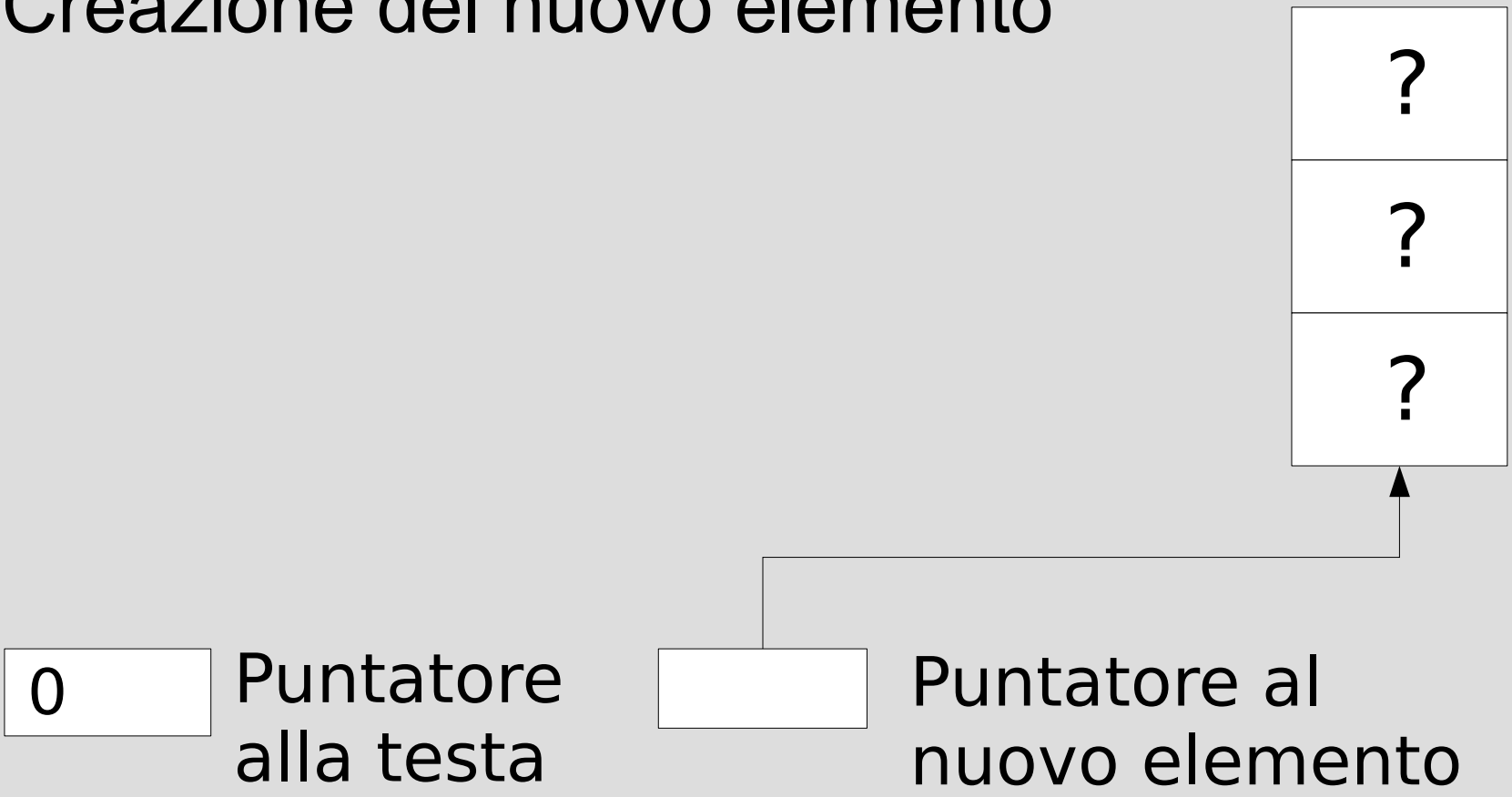
Inserimento in testa

- **Algoritmo per inserire un elemento in testa ad una lista doppia**
- Struttura dati necessaria

- Partiamo dall'ipotesi di lista di partenza vuota

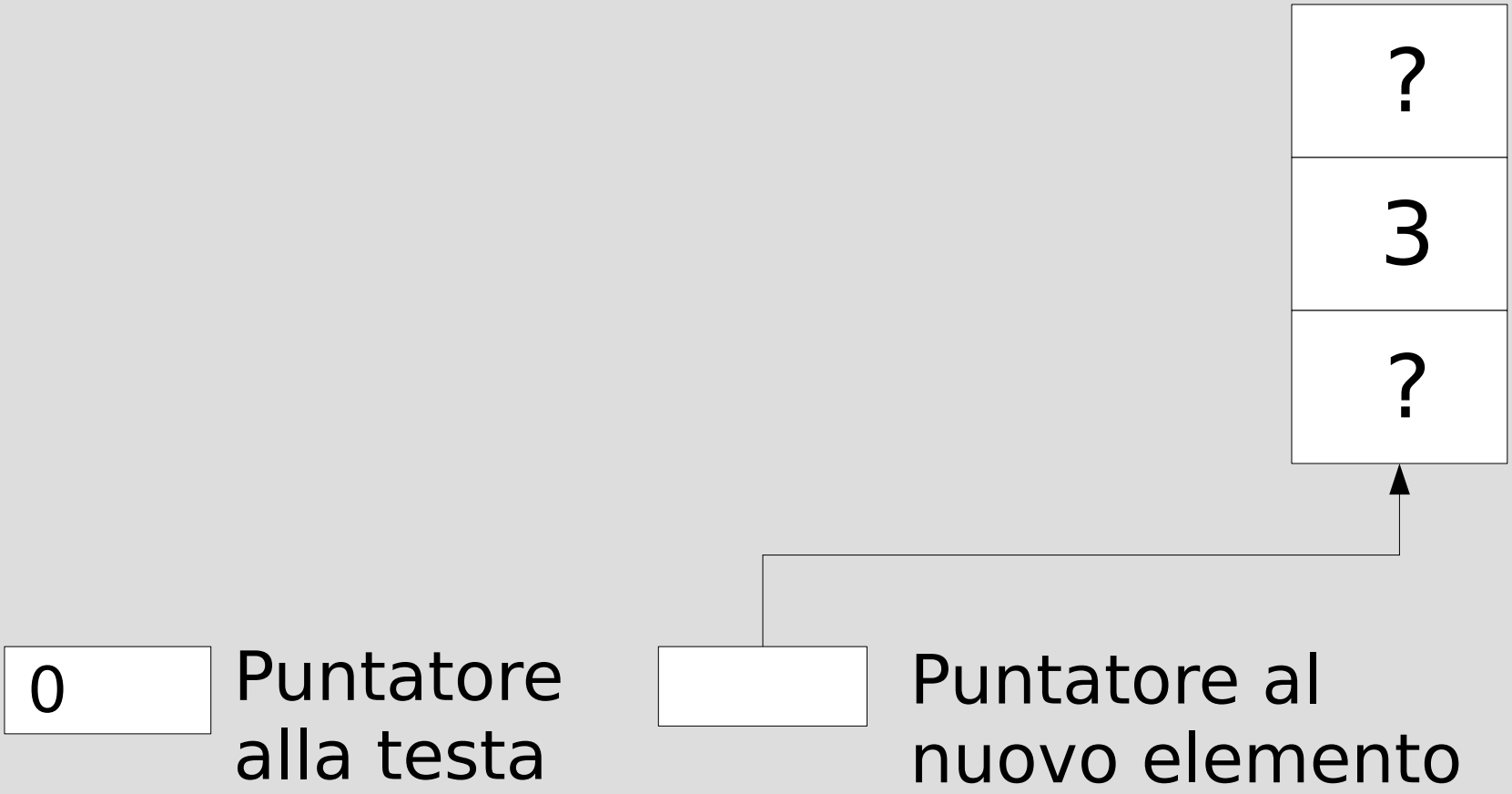
Inserimento in testa

- Lista di partenza vuota
- Creazione del nuovo elemento



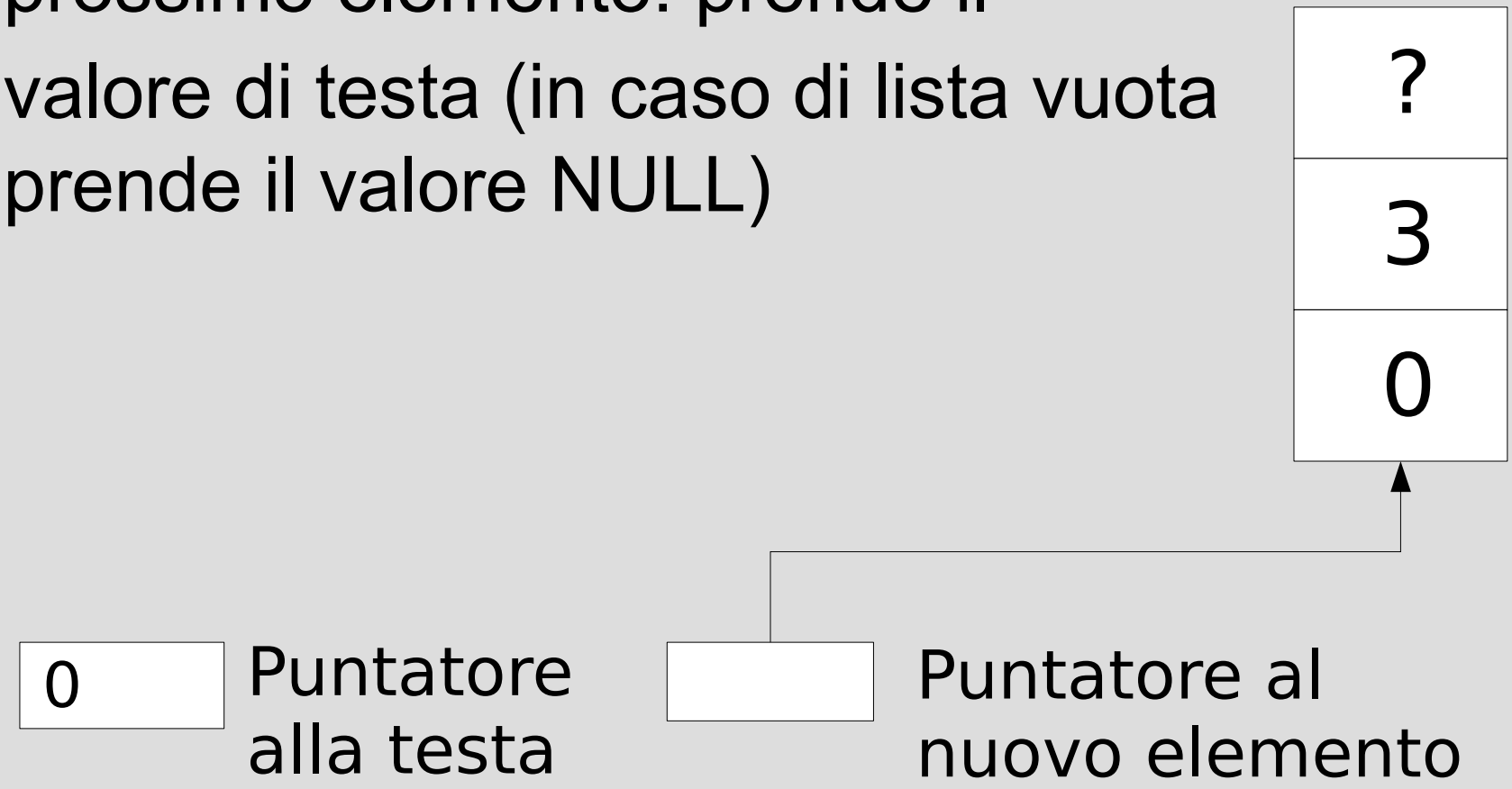
Inserimento in testa

- Inizializzazione campo informazione



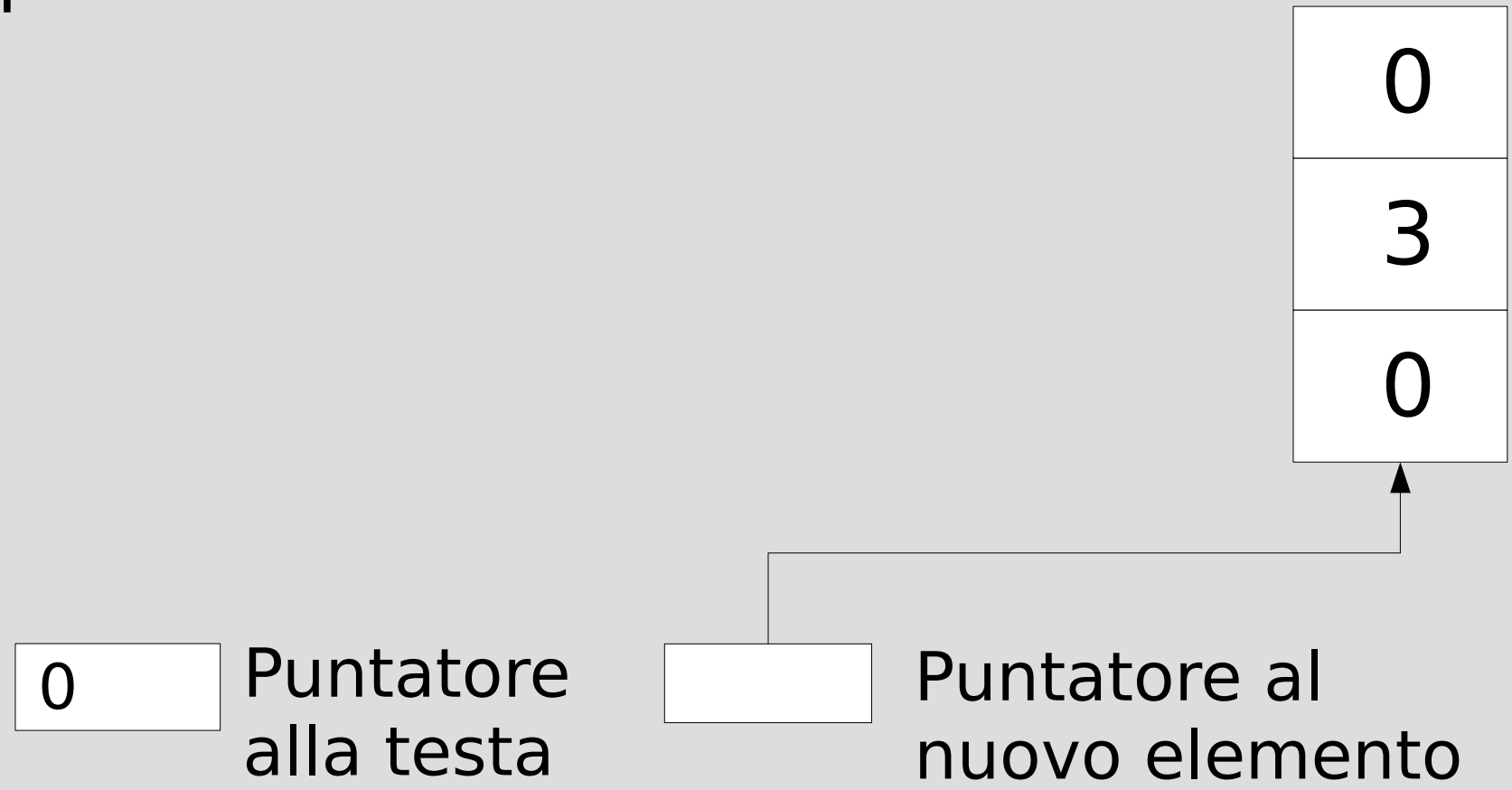
Inserimento in testa

- Inizializzazione puntatore al prossimo elemento: prende il valore di testa (in caso di lista vuota prende il valore NULL)



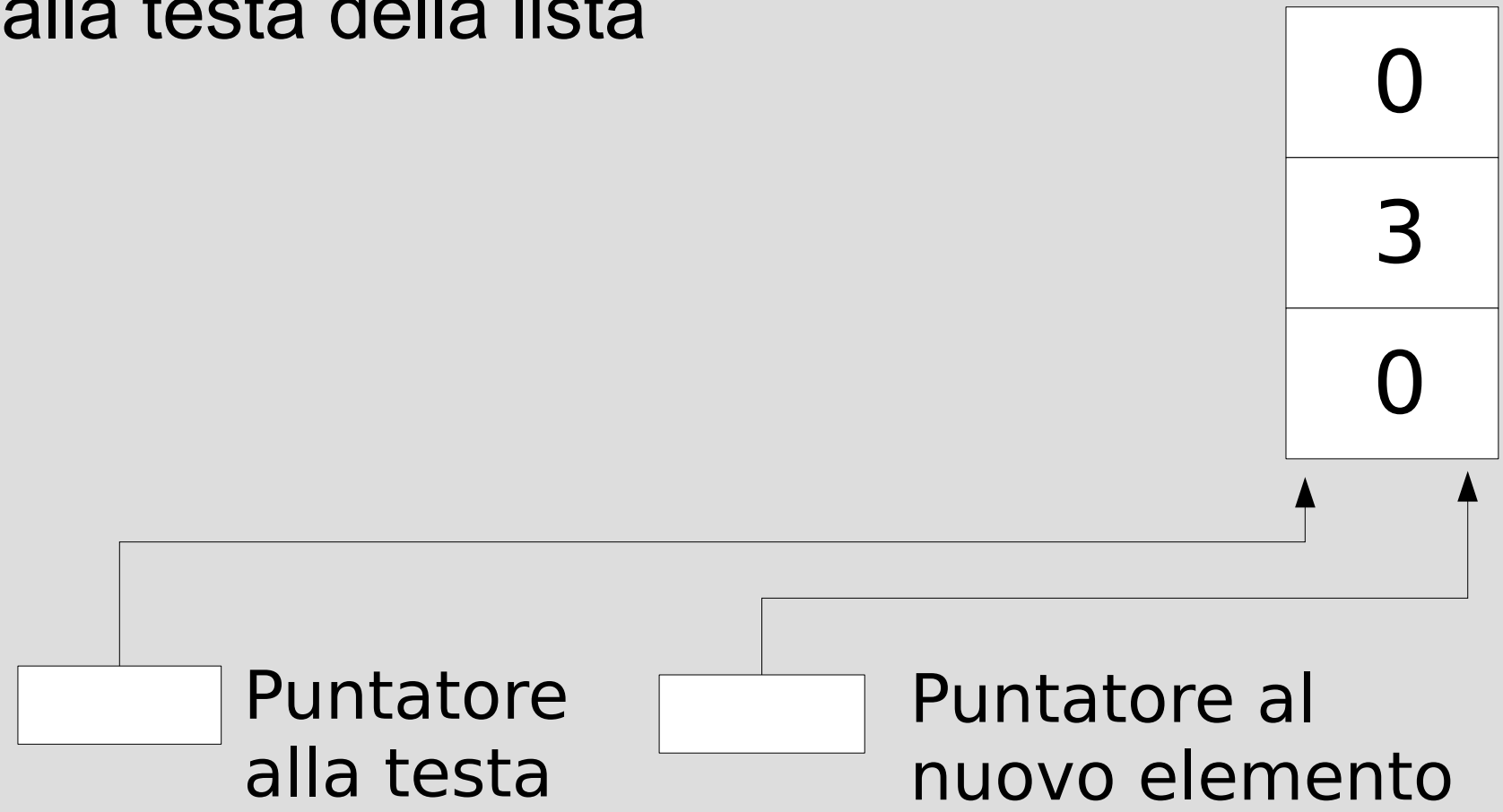
Inserimento in testa

- Inizializzazione puntatore al precedente elemento



Inserimento in testa

- Aggiornamento puntatore alla testa della lista

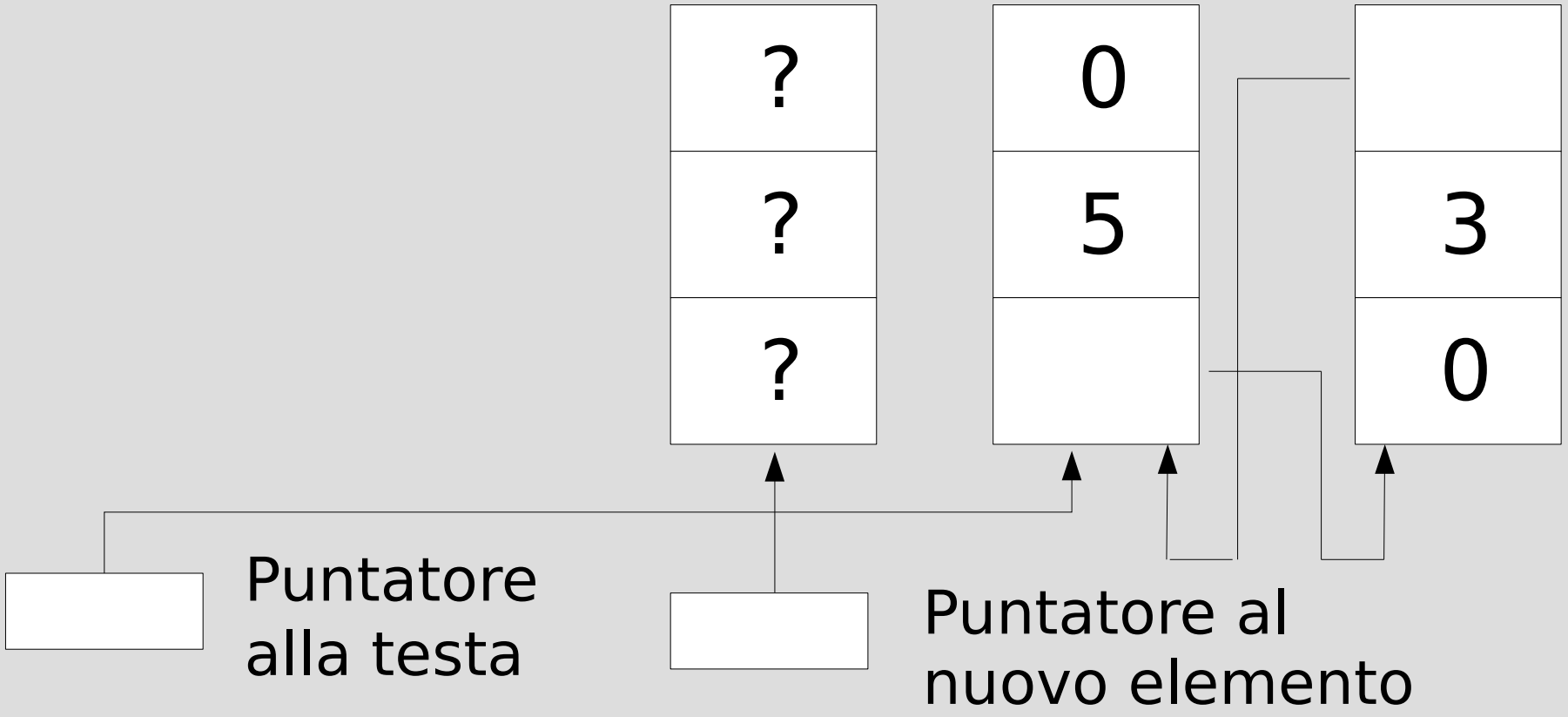


Inserimento in testa

- Caso lista non vuota
- Lista prima dell'invocazione composta di due elementi

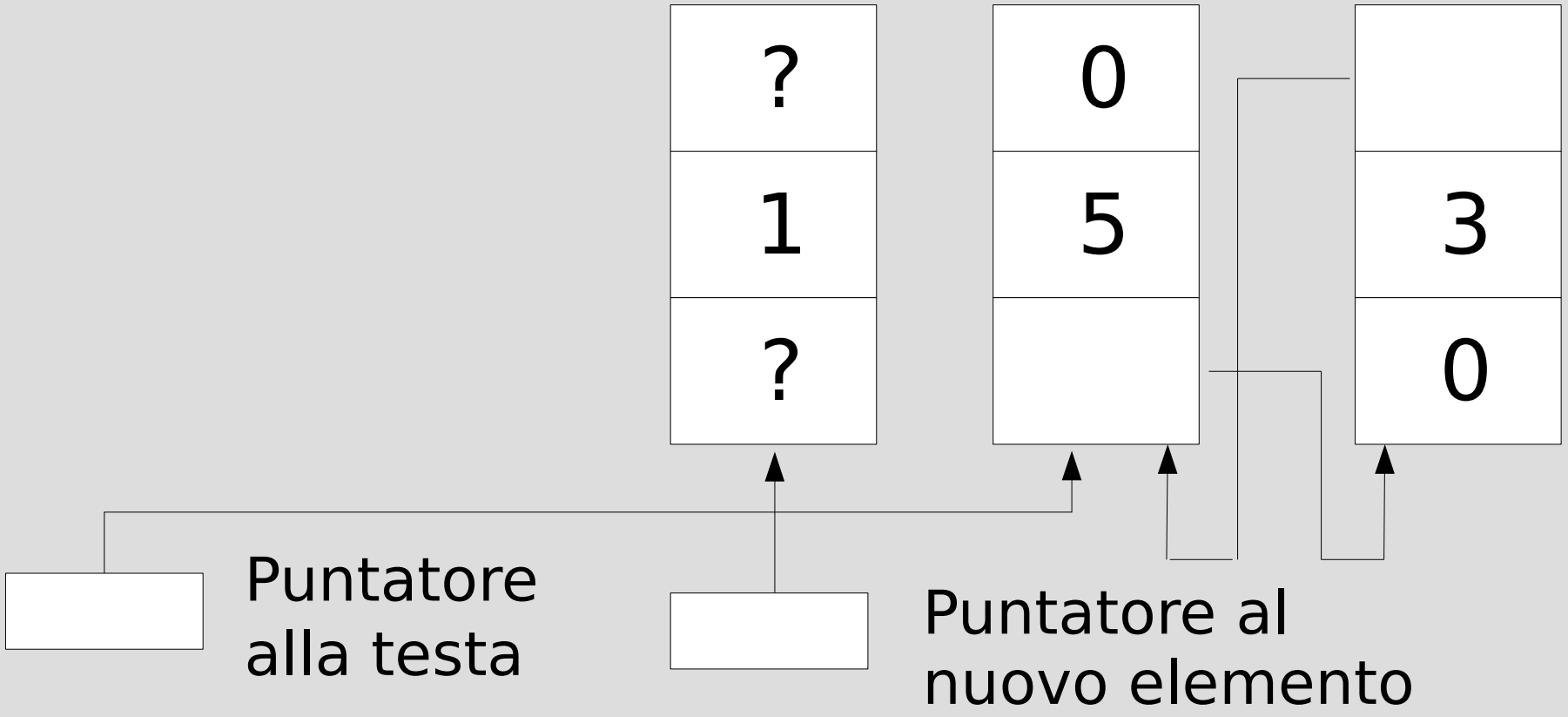
Inserimento in testa

- Creazione del nuovo elemento



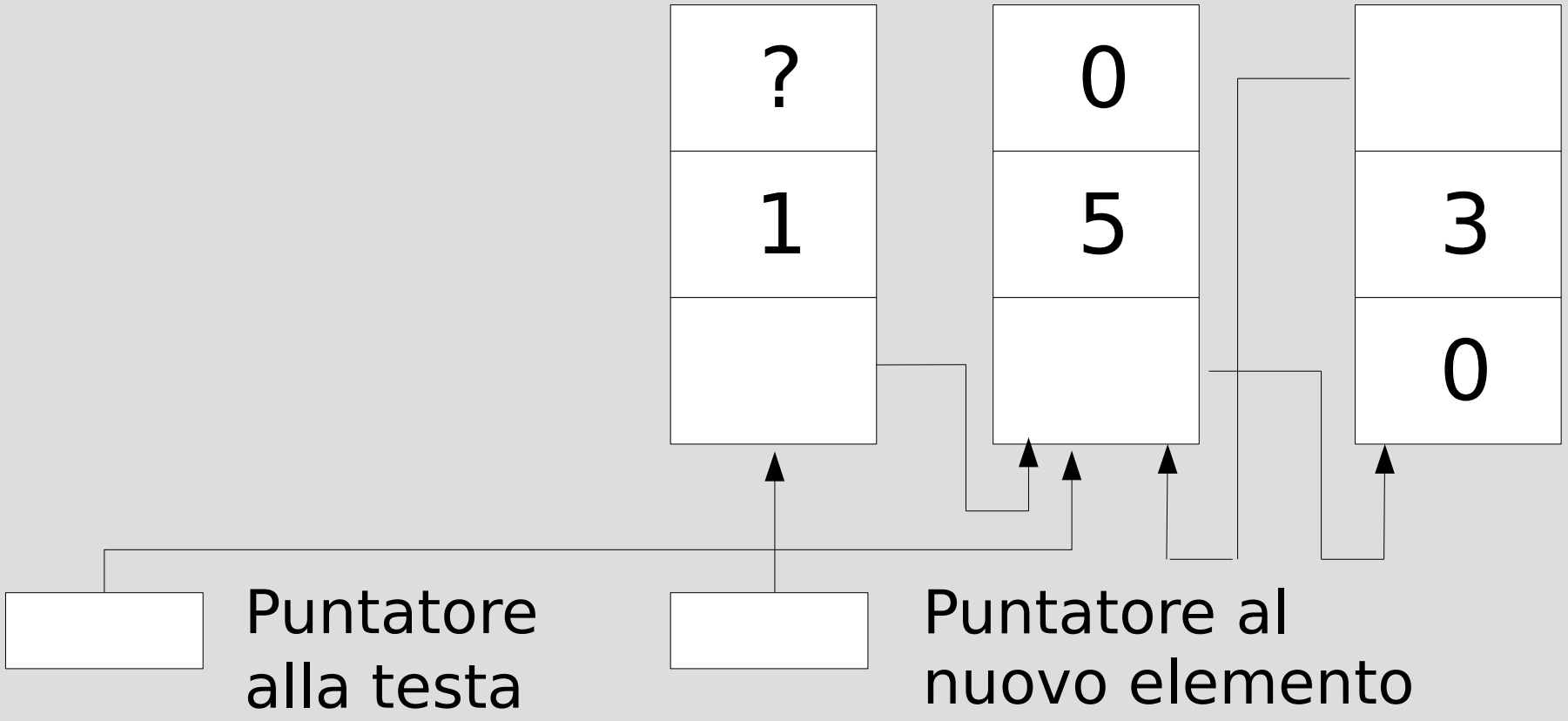
Inserimento in testa

- Inizializzazione campo informazione



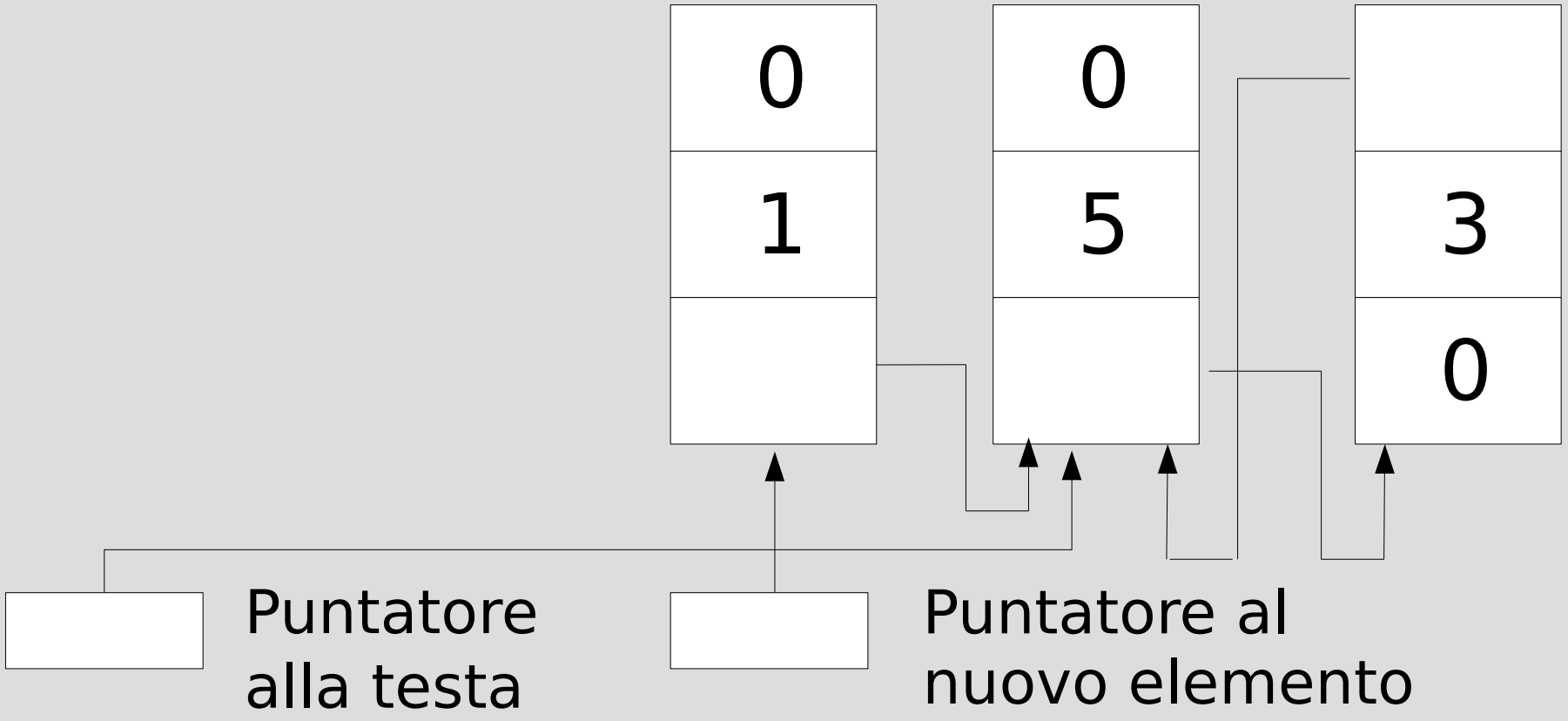
Inserimento in testa

- Inizializzazione *puntatore al prossimo* elemento (prende il valore di testa)



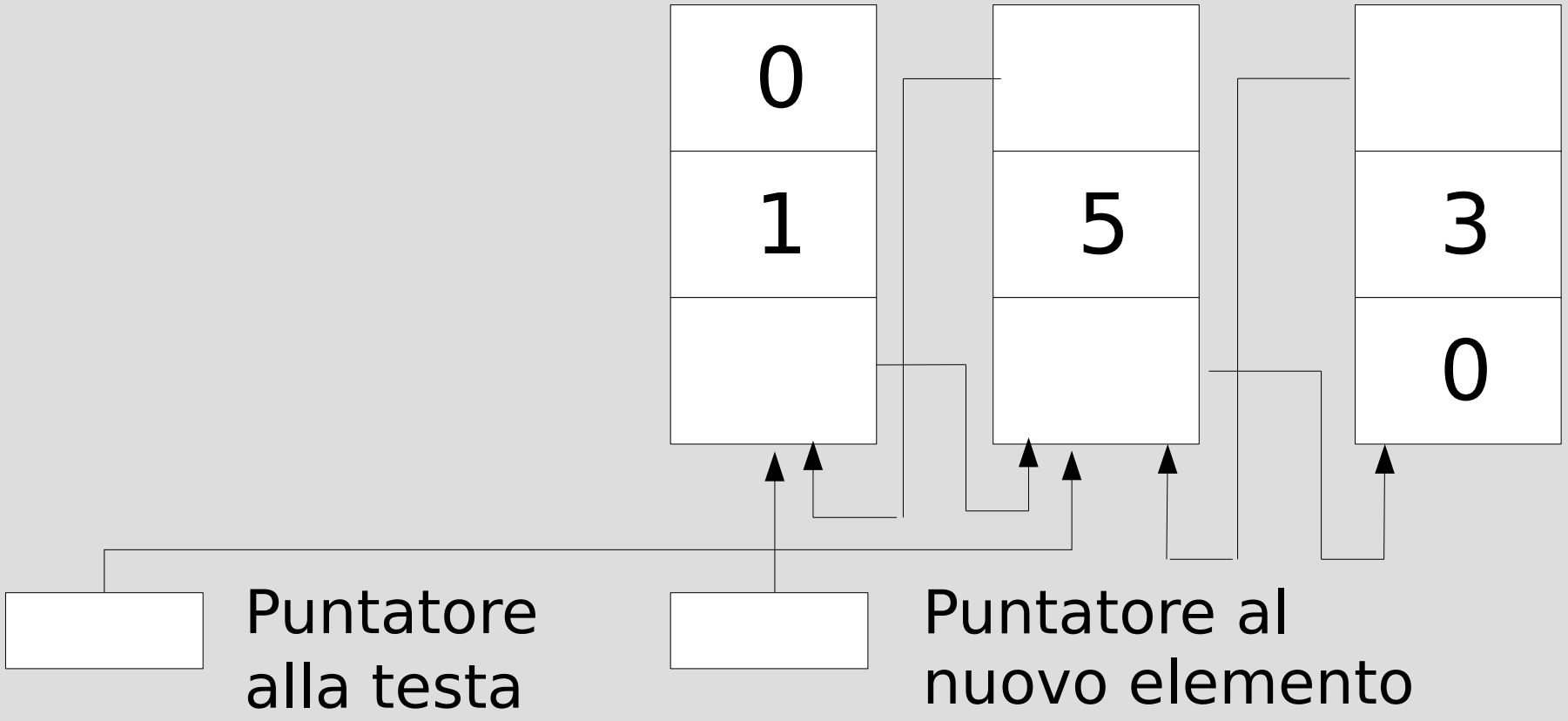
Inserimento in testa

- Inizializzazione *puntatore al precedente* elemento



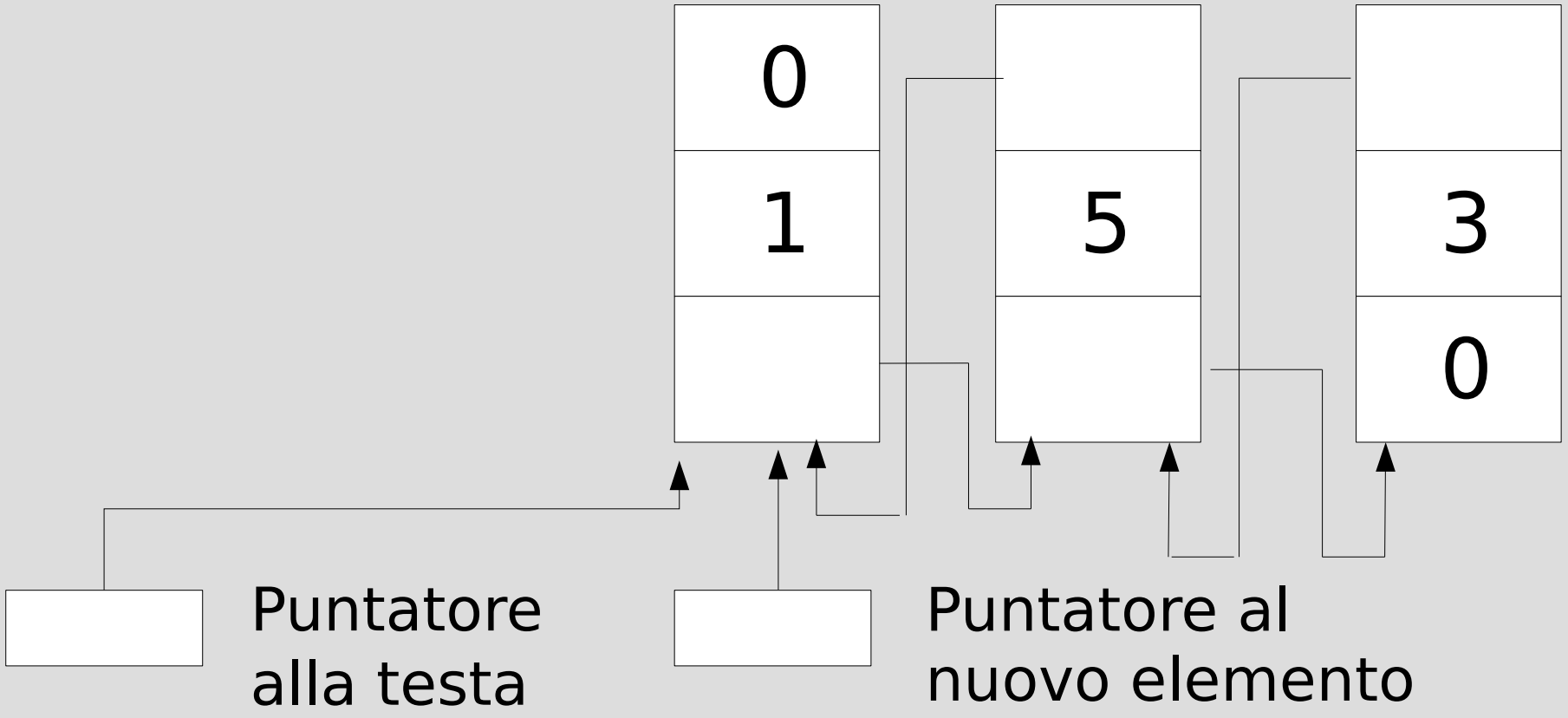
Inserimento in testa

- Aggiornamento *puntatore a elemento precedente* nell'ex primo elemento della lista



Inserimento in testa

- Aggiornamento *puntatore alla testa* della lista



Algoritmo

- Creazione elemento
- Inizializzazione campo informazione
- Aggancio dell'elemento alla lista
  - Inizializzazione del campo *puntatore al prossimo* con l'indirizzo della testa
  - Inizializzazione del campo *puntatore al precedente* con 0
  - Se la lista non era vuota, aggiornamento del *puntatore al precedente* nell'ex primo elemento
  - Aggiornamento del *puntatore alla testa* della lista

Programma

- Realizzare la funzione *inserisci_in_testa* su una lista doppia
  - Si parta da *lista_doppia_solo_main.cc*
  - Supporre che la funzione *inserisci_in_testa* abbia tipo di ritorno *void*
  - Funzioni *stampalista* ed *eliminalista* già presenti

Estrazione per valore

- Estrazione di un elemento per valore da una lista doppia
  - Valore passato come parametro d'ingresso
- Algoritmo generale:

1) Ricerca dell'elemento da estrarre
2) Sgancio dell'elemento dalla lista
3) Deallocazione dell'elemento

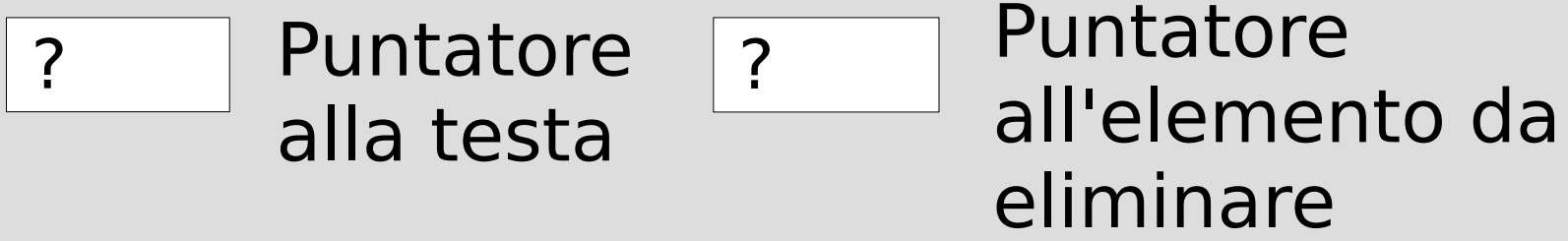
Puntatori ausiliari

- Per inserimenti/estrazioni per valore in liste semplici si rendevano necessari due puntatori ausiliari
- Sono necessari anche con le liste doppie?

No, non sono necessari grazie alla presenza del puntatore all'elemento precedente

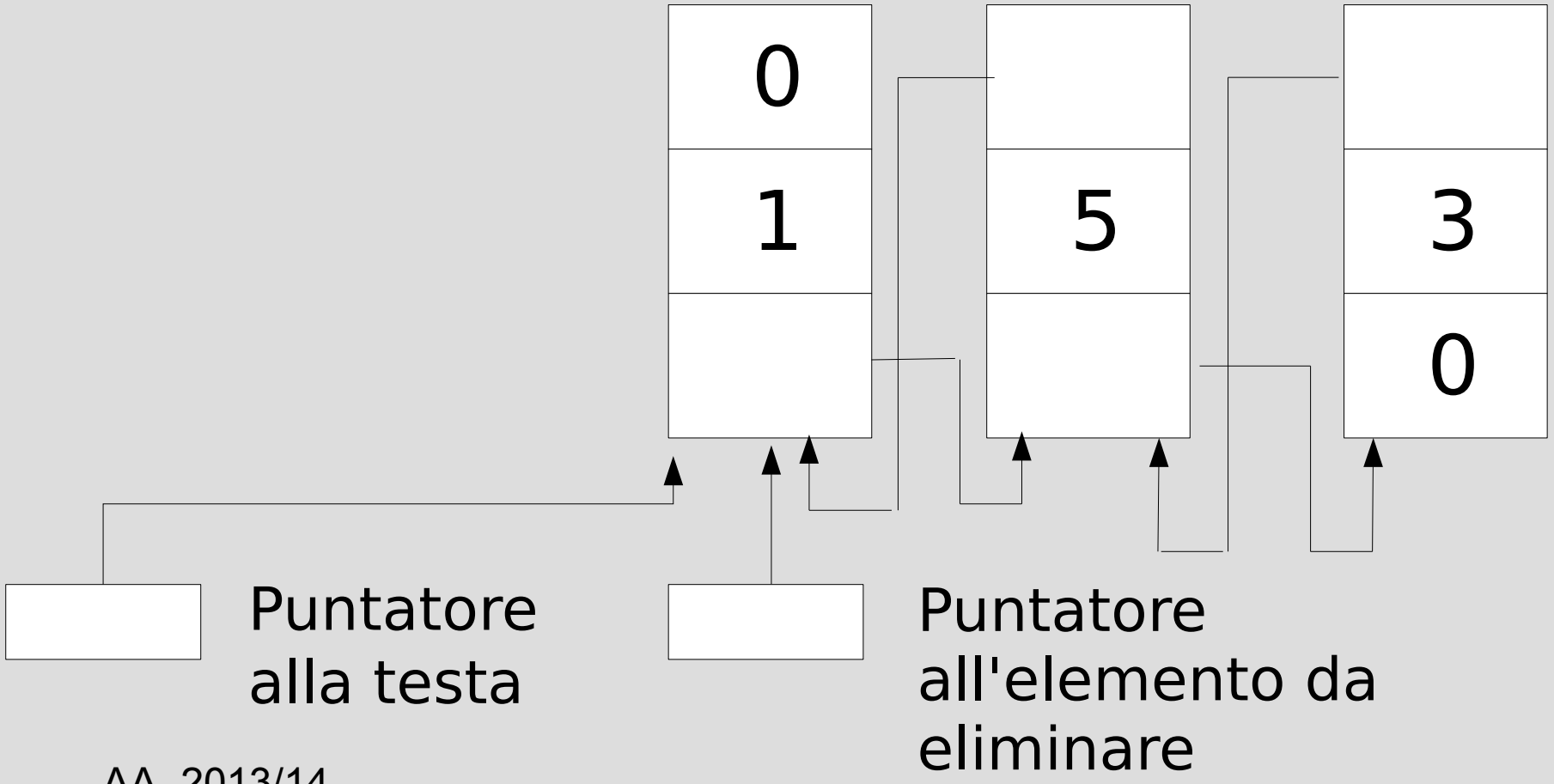
Estrazione per valore

- Si supponga di voler estrarre l'elemento di valore 5 da una lista doppia
- Si analizzi il caso in cui la lista contenga più di un elemento
- Struttura dati necessaria



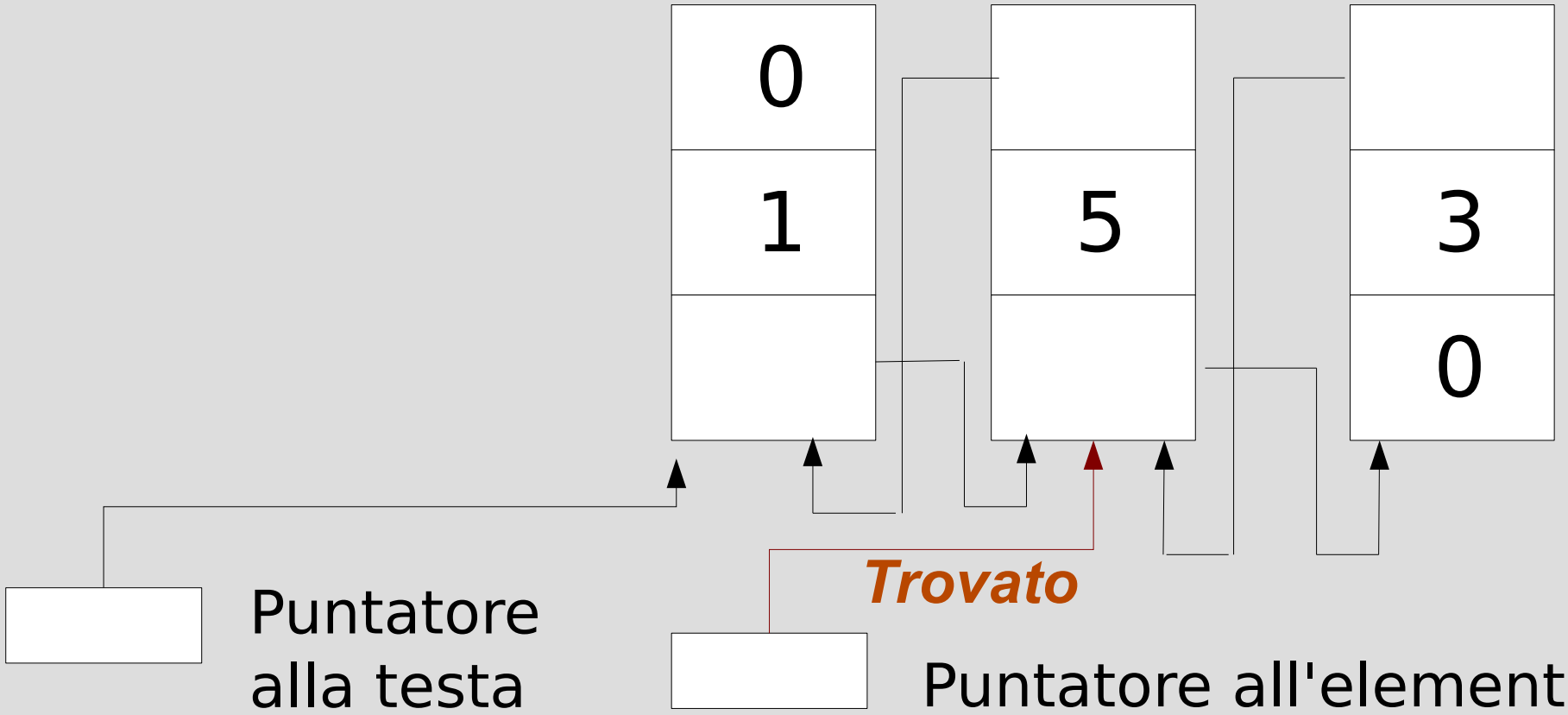
Estrazione per valore

- Ricerca elemento da estrarre (di valore 5)



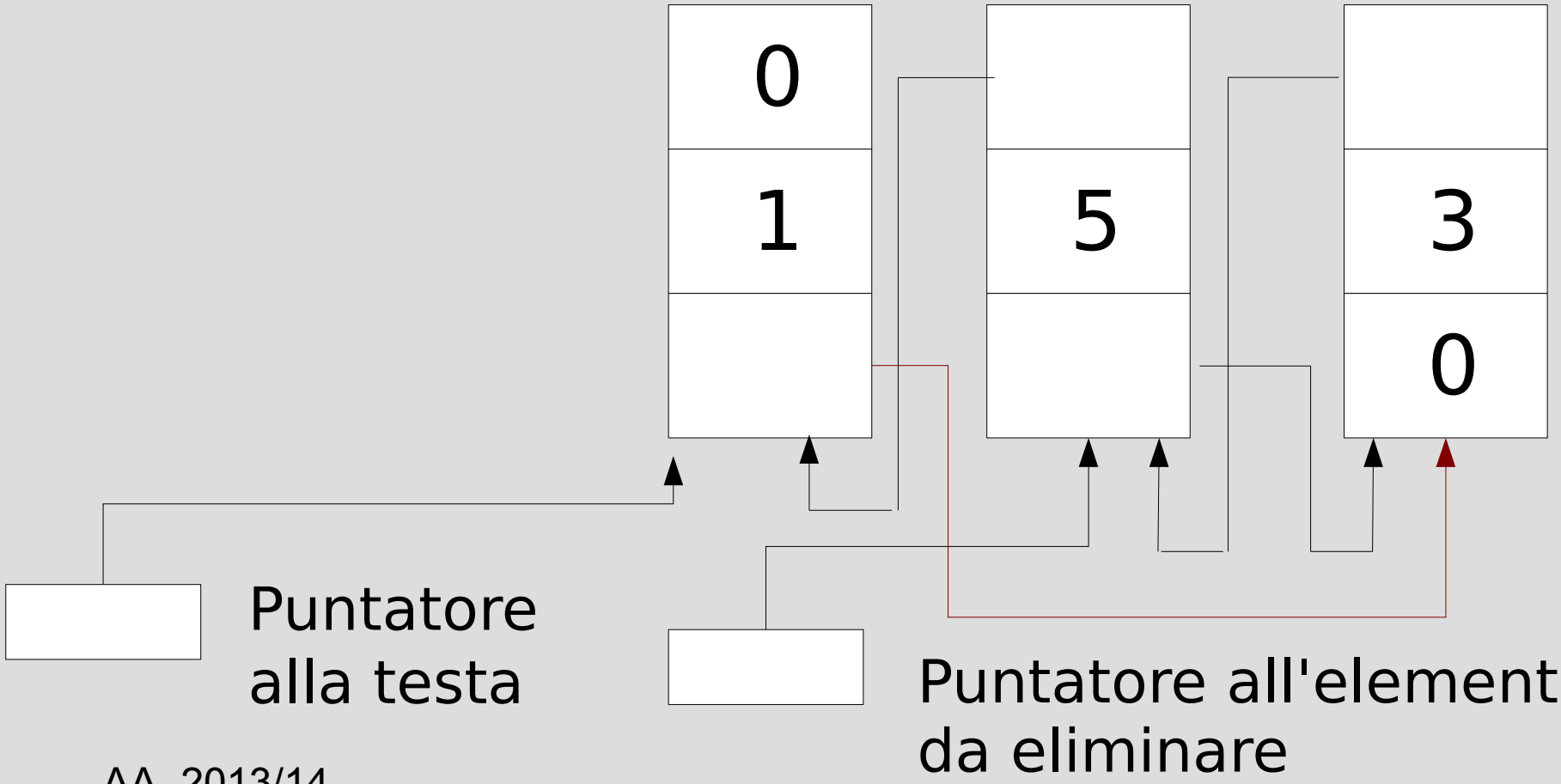
Estrazione per valore

- Ricerca elemento da estrarre (di valore 5)



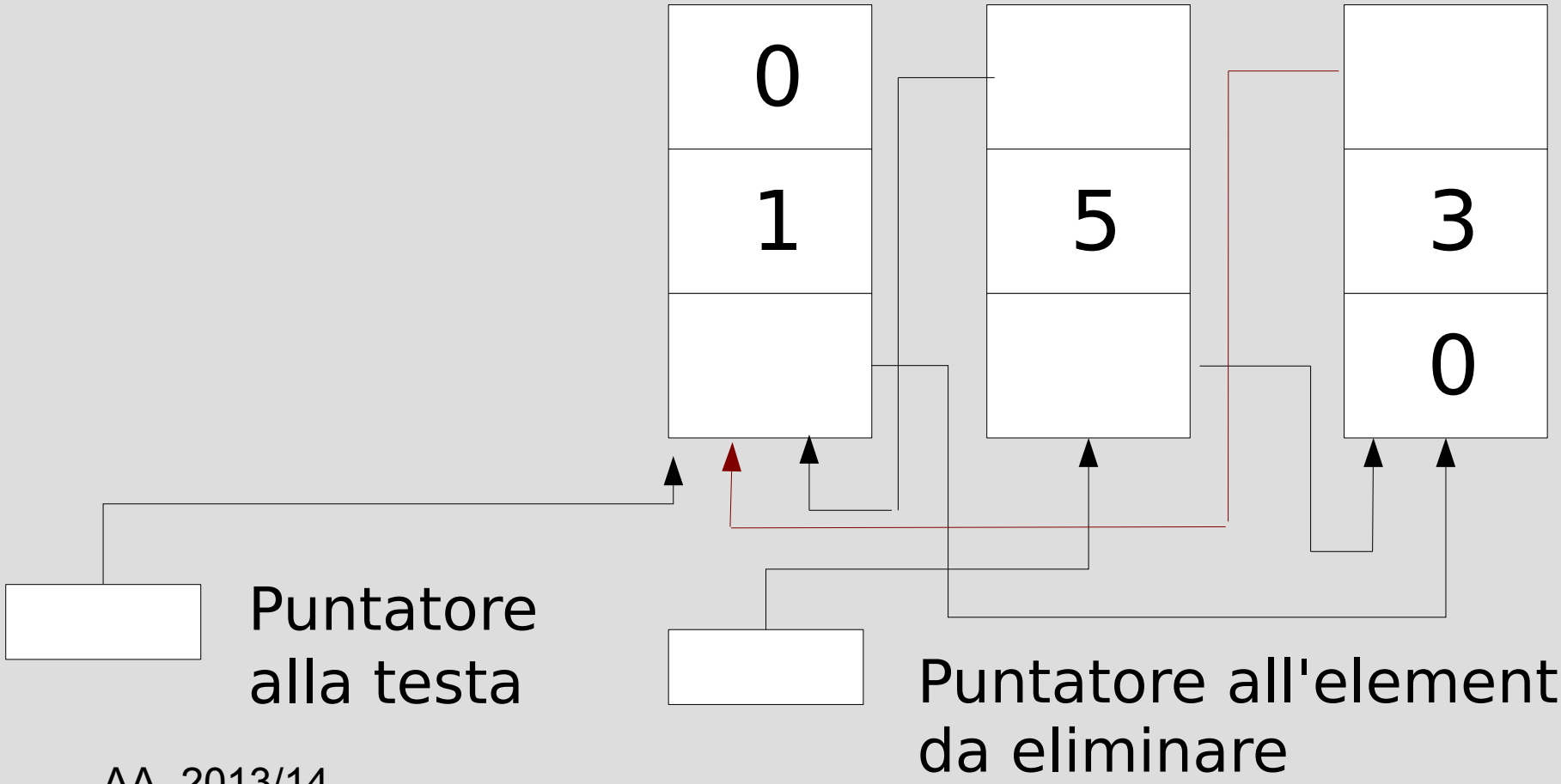
Estrazione per valore

- Aggiornamento campo *puntatore al prossimo* nell'elemento precedente



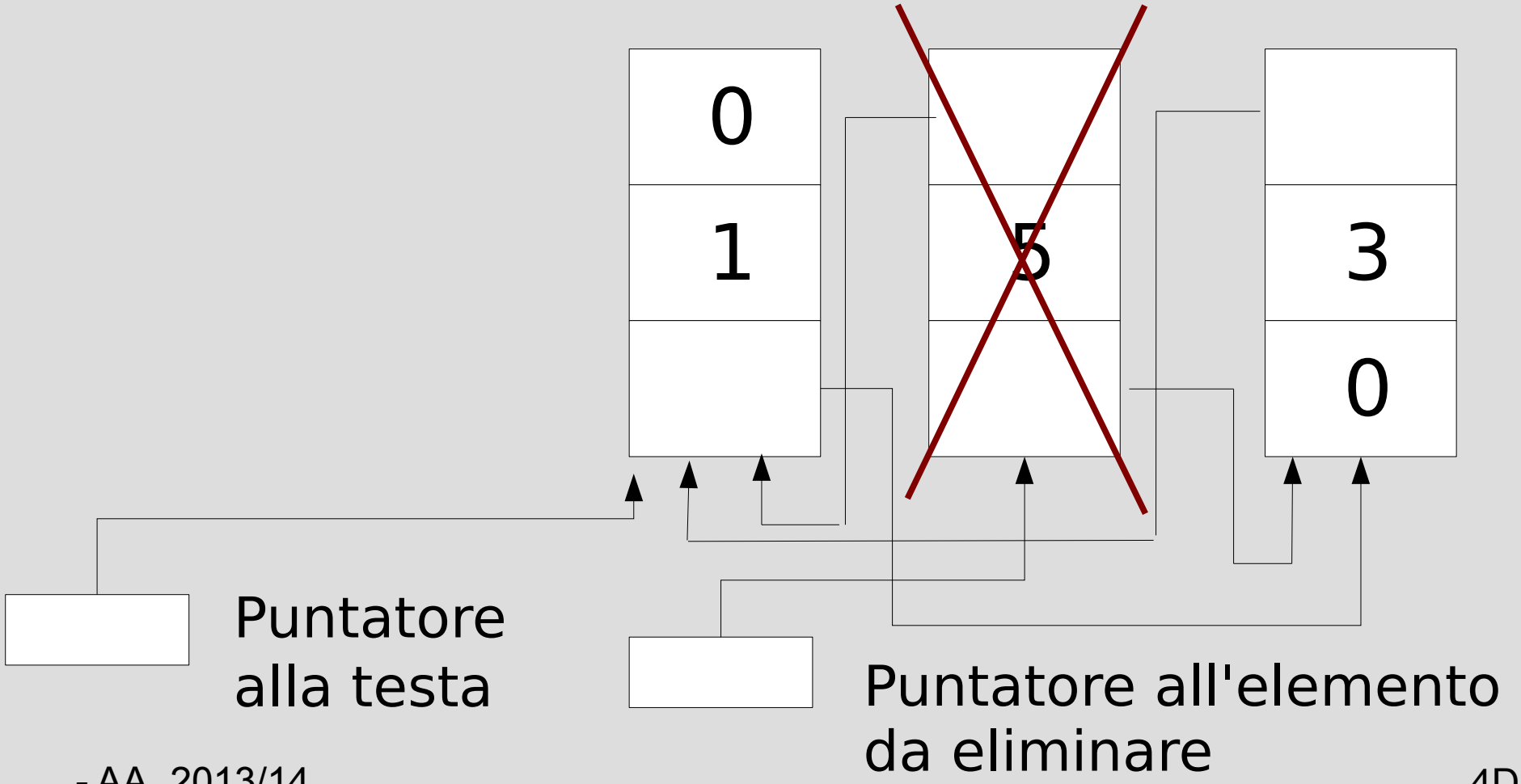
Estrazione per valore

- Aggiornamento campo *puntatore al precedente* nell'elemento successivo



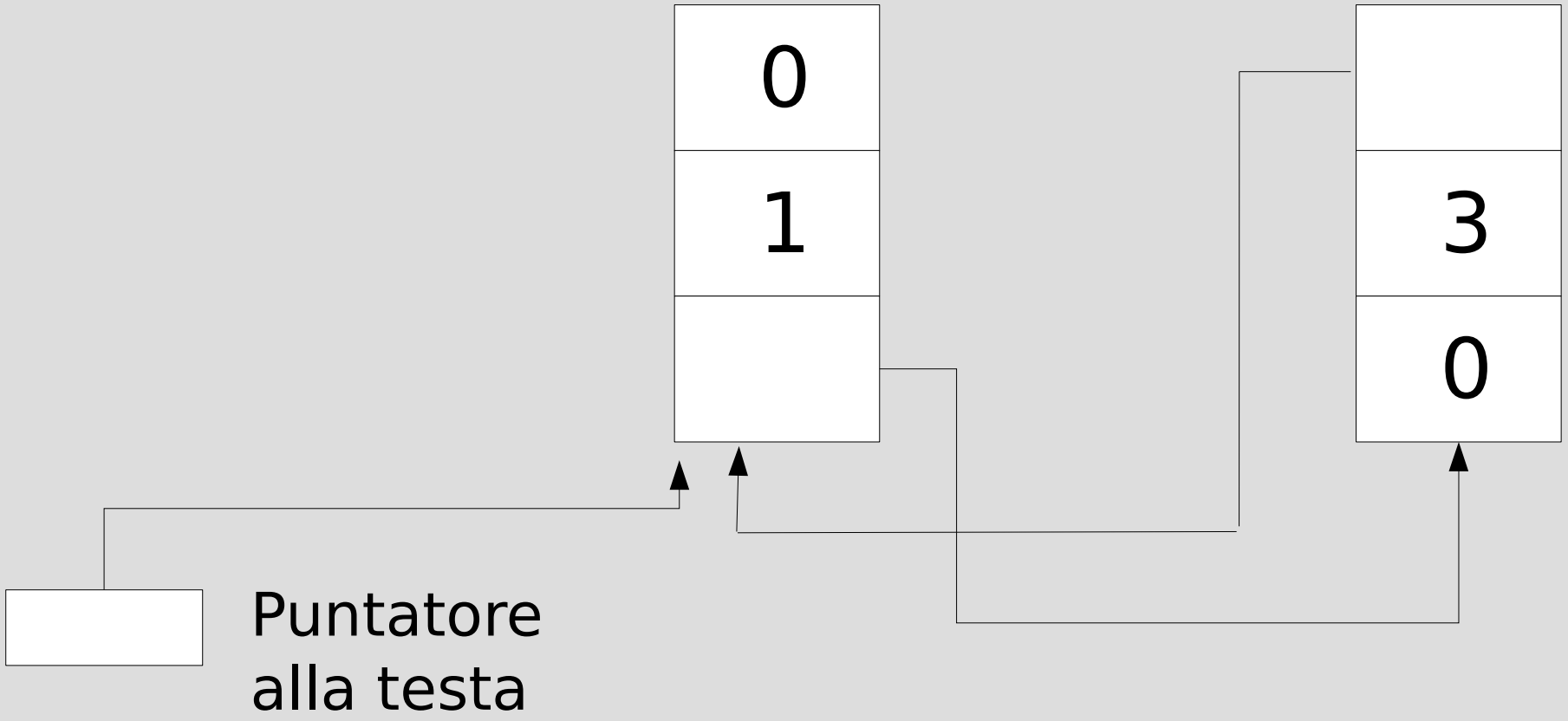
Estrazione per valore

- Deallocazione dell'elemento



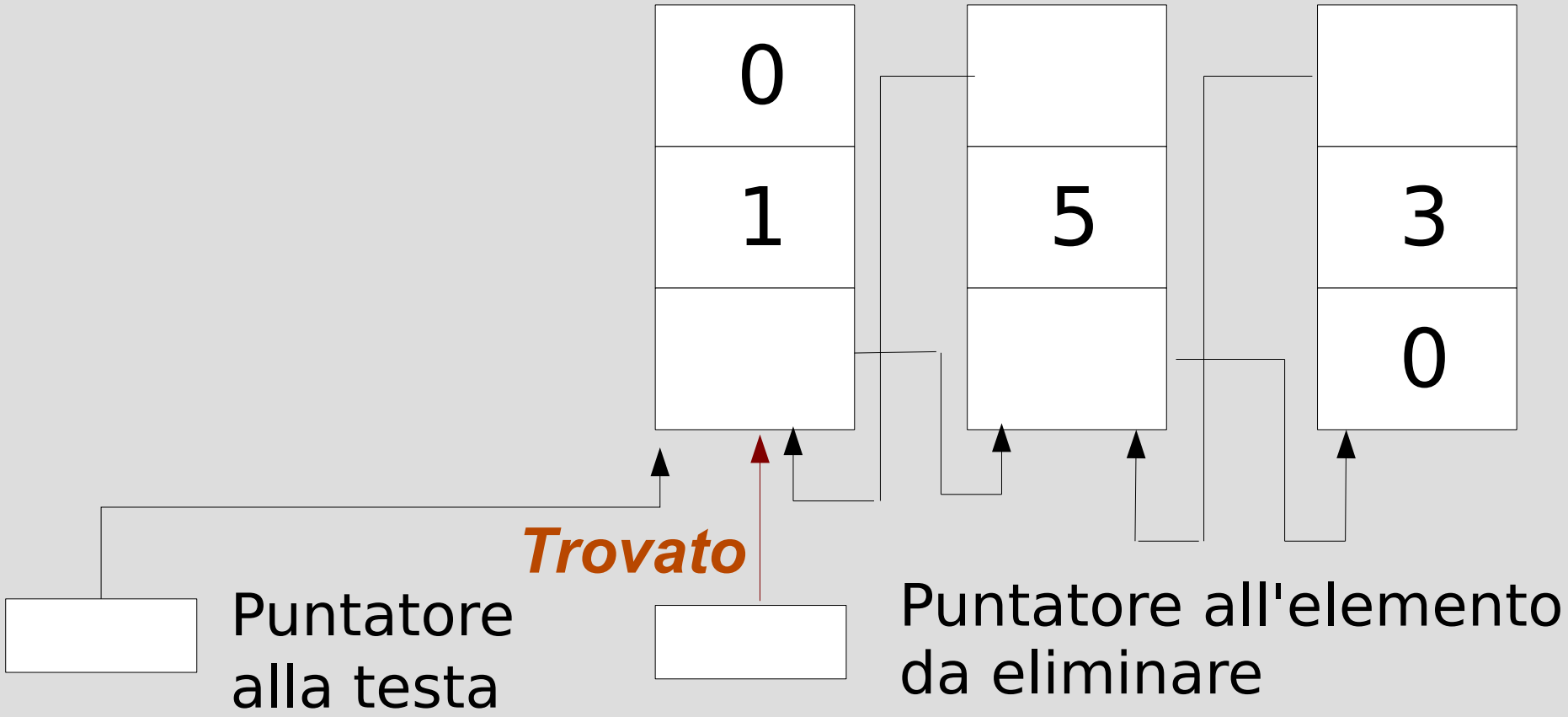
Estrazione per valore

- Lista al termine della estrazione



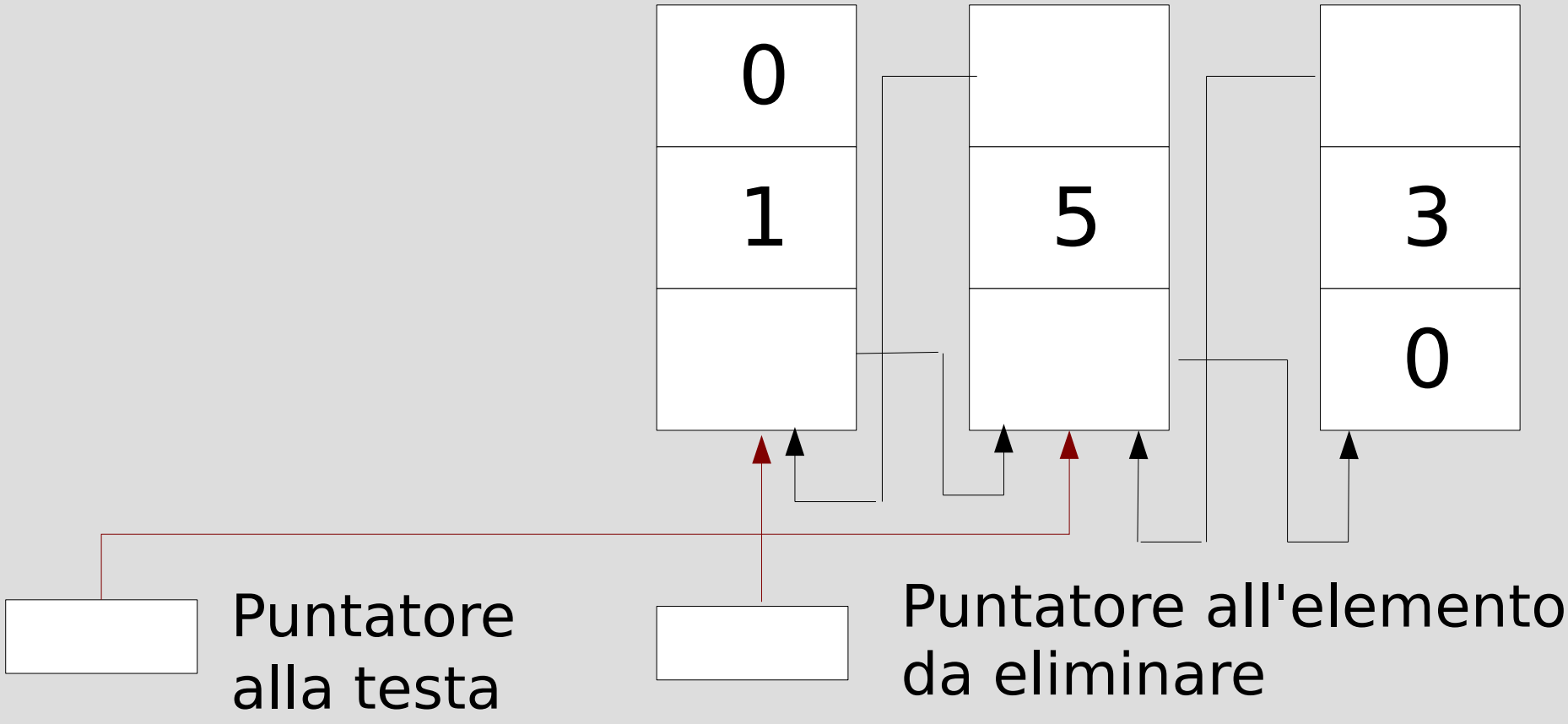
Estrazione dalla testa

- Caso in cui l'elemento da eliminare sia il primo della lista (valore 1)



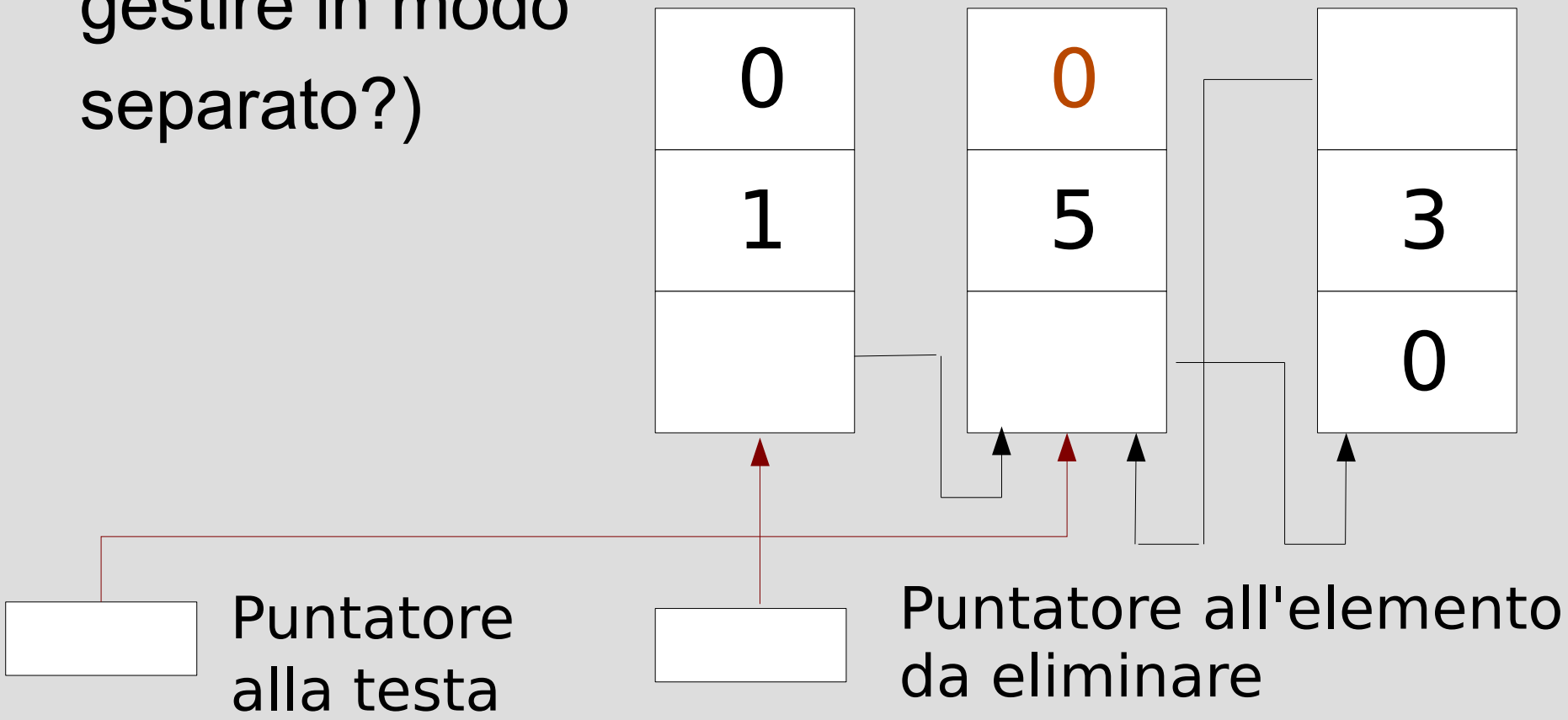
Estrazione dalla testa

- Aggiornamento del puntatore alla testa della lista



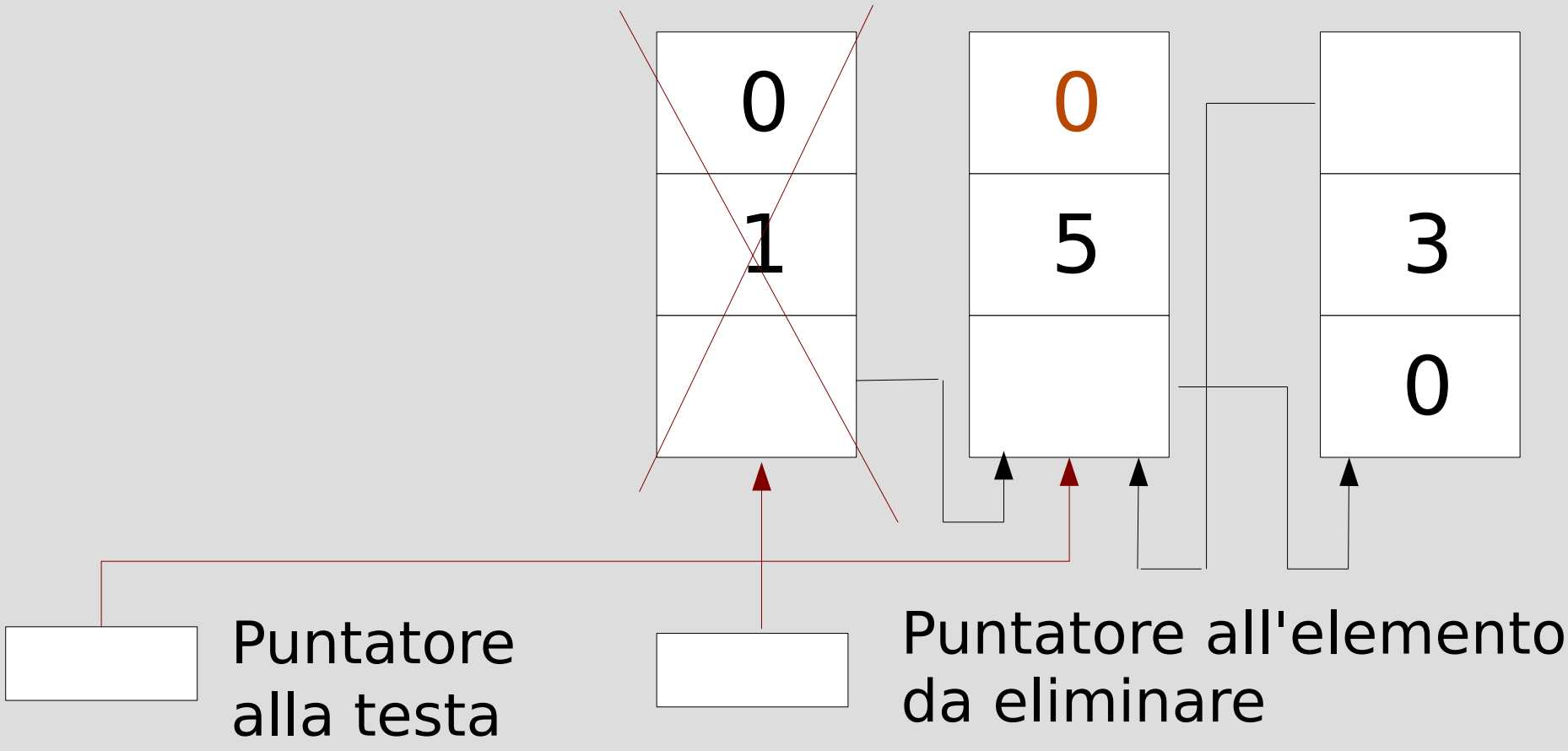
Estrazione dalla testa

- Aggiornamento del campo *puntatore al precedente* nell'elemento successivo (da gestire in modo separato?)



Estrazione dalla testa

- Deallocazione dell'elemento

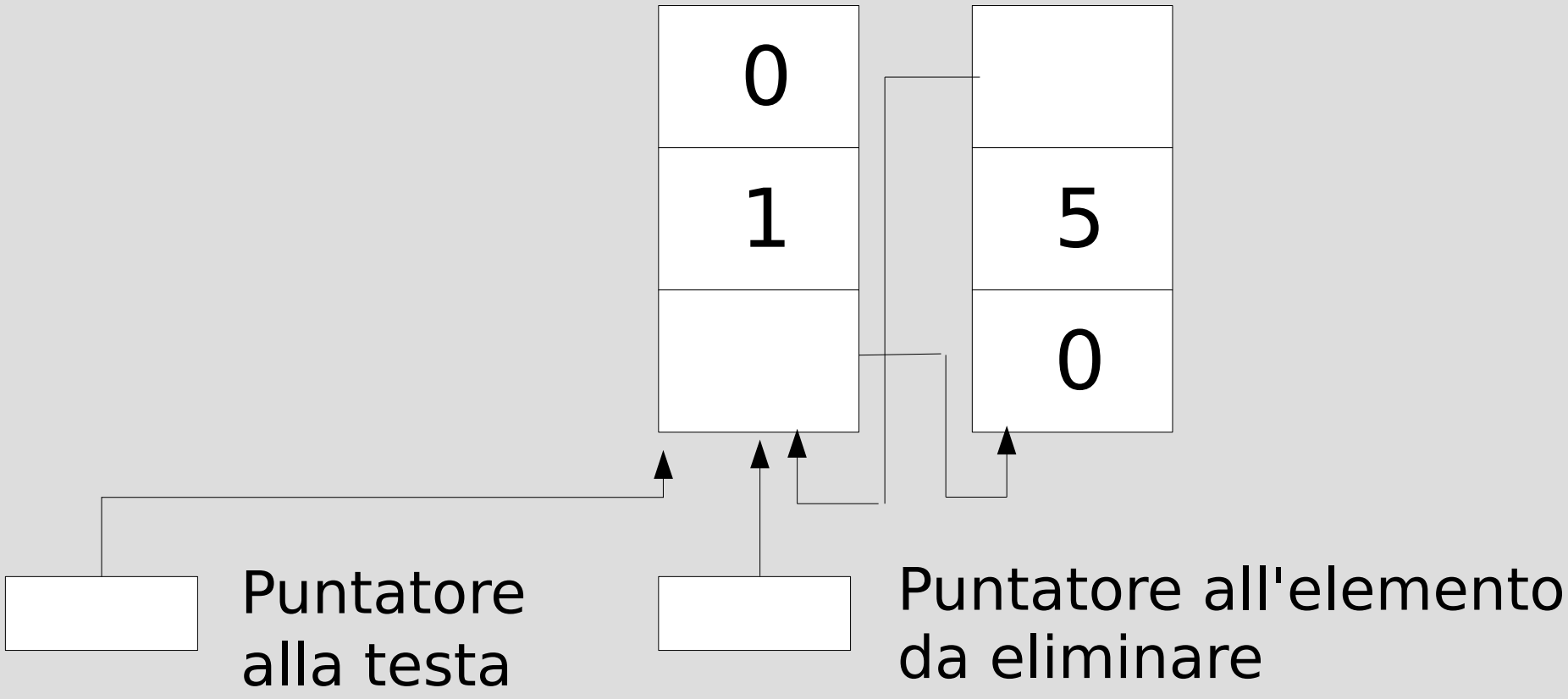


Estrazione dalla testa

- Lista al termine della funzione

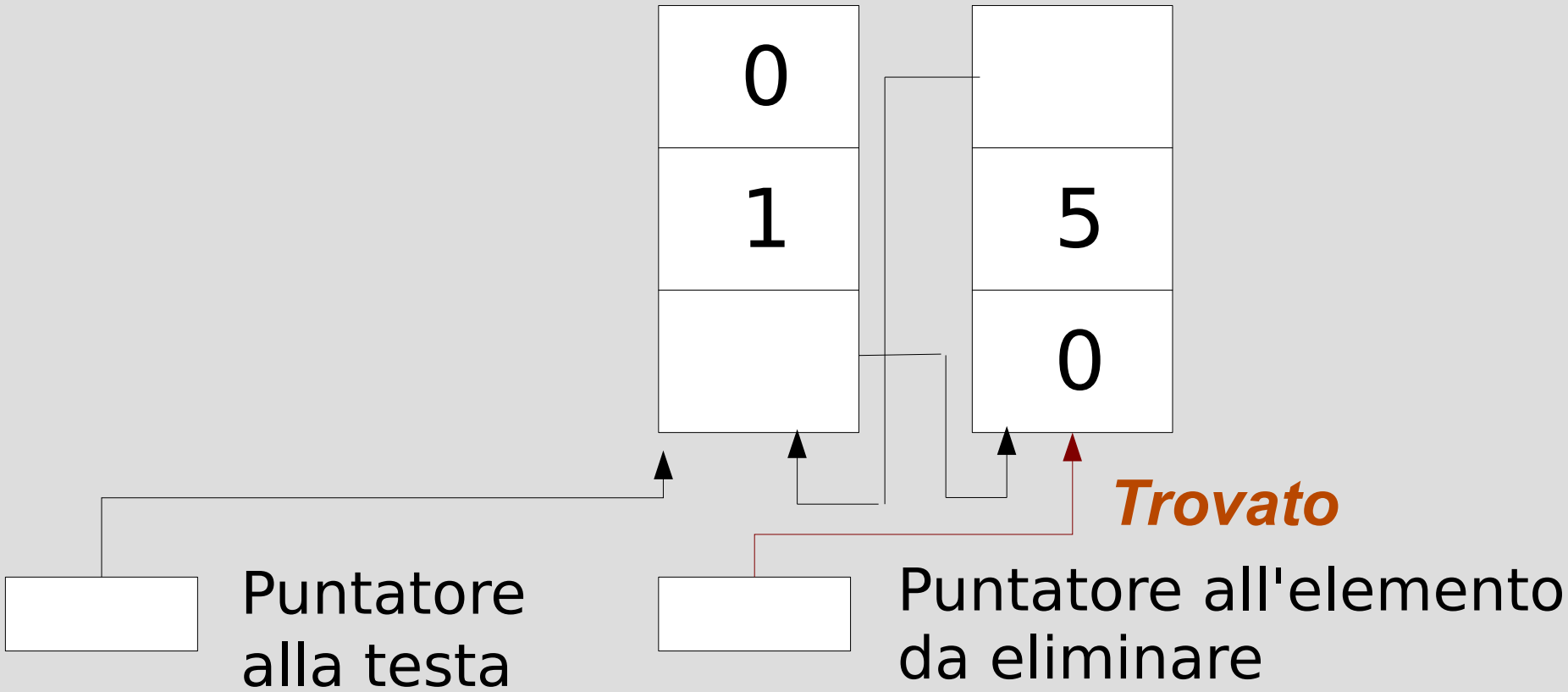
Estrazione dalla coda

- Caso in cui capita di estrarre l'ultimo elemento – la coda alla lista (valore 5)



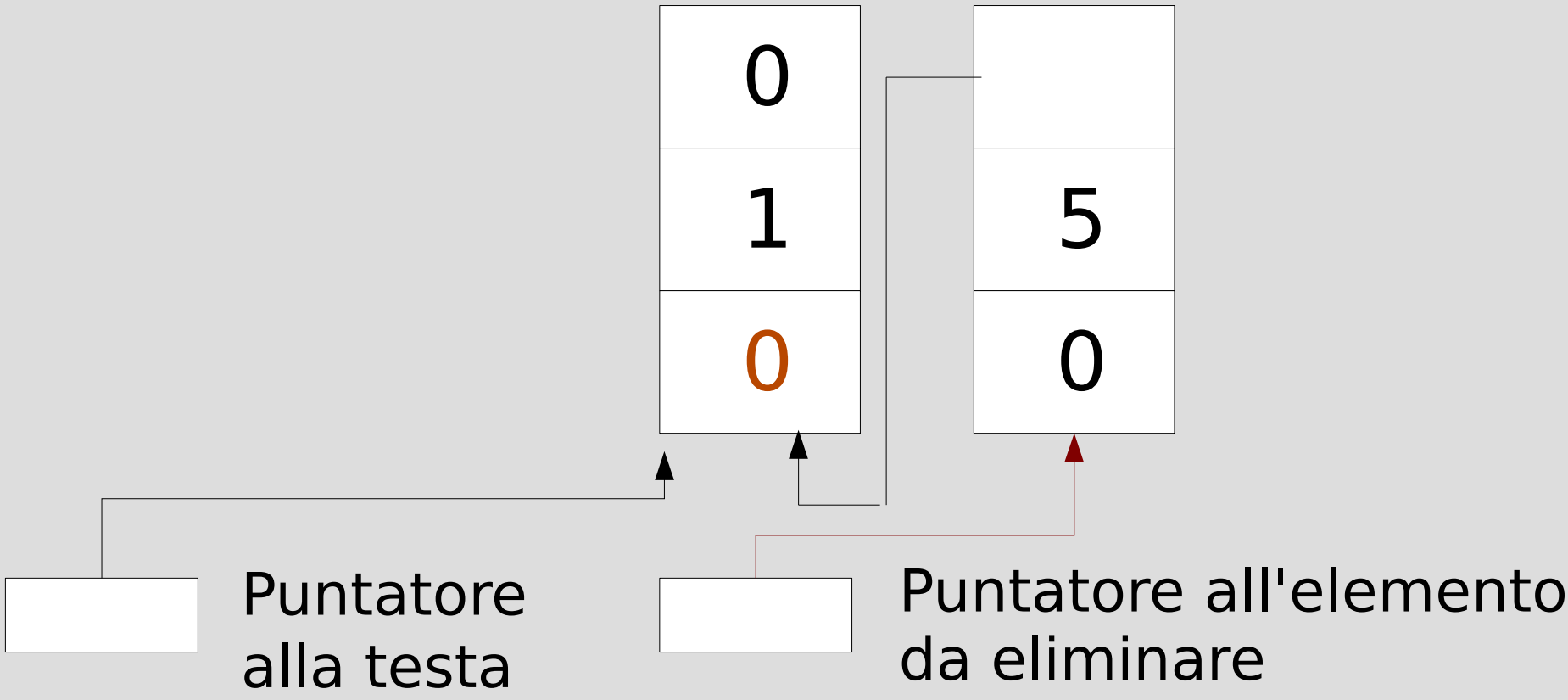
Estrazione dalla coda

- Ricerca elemento da estrarre



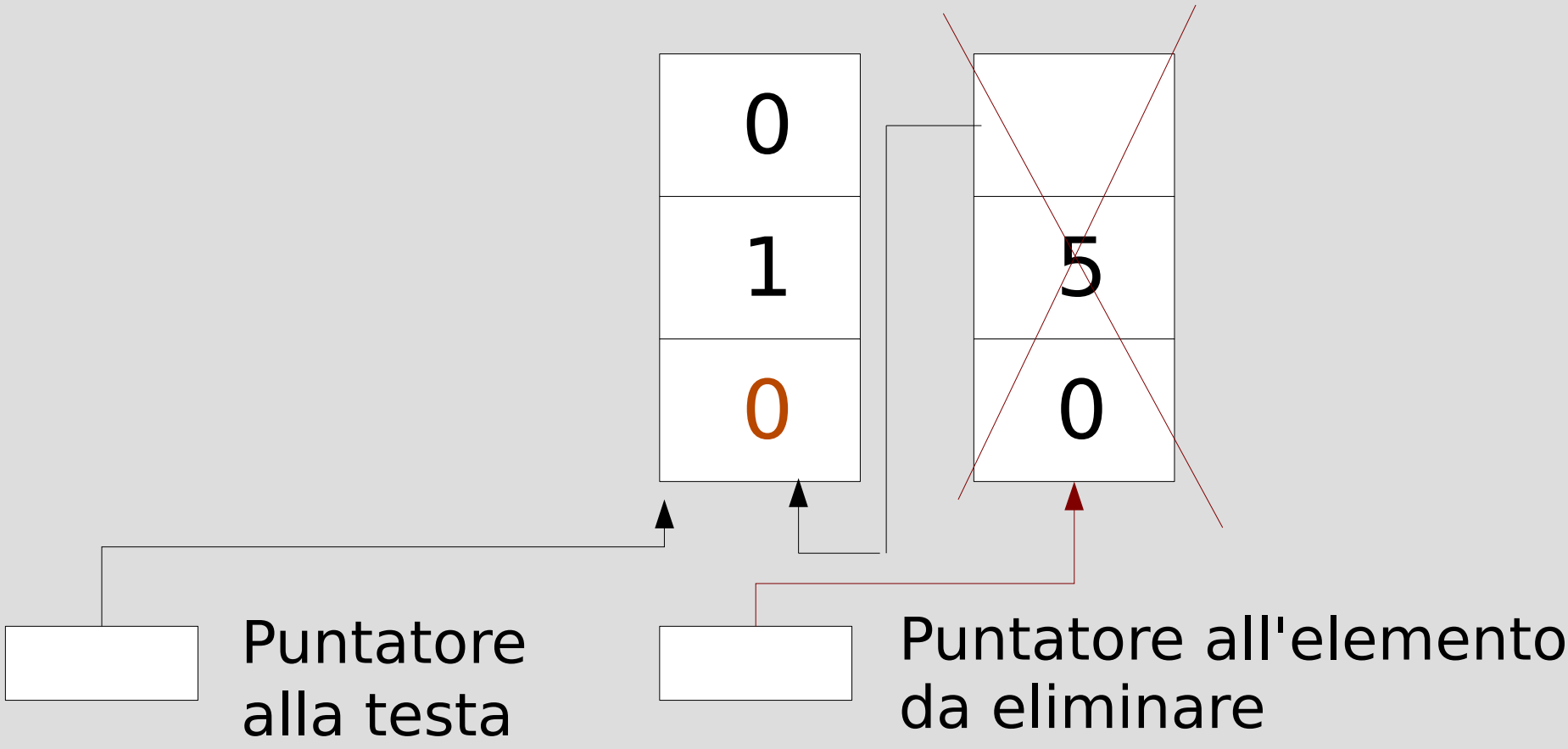
Estrazione dalla coda

- Aggiornamento campo *puntatore al prossimo* nell'elemento precedente



Estrazione dalla coda

- Deallocazione dell'elemento



Estrazione dalla coda

- Lista al termine della funzione

Algoritmo generale: estrazione

- Ricerca dell'elemento da estrarre
- Sgancio dell'elemento dalla lista
  - Controllo che l'elemento sia o meno la testa
    - Se l'elemento è la testa, aggiornamento del *puntatore alla testa* della lista
    - Altrimenti, aggiornamento del *puntatore al prossimo* nell'elemento precedente
  - Controllo che l'elemento sia o meno la coda
    - Se non è la coda, aggiornamento del *puntatore al precedente* nell'elemento successivo
- Deallocazione dell'elemento

Programma

Realizzare la funzione *estrai_per_valore* in una lista doppia

- Supporre che la *estrai_per_valore* abbia tipo di ritorno *bool*
  - *Torna false se il valore non è presente nella lista*
  - *Torna true se il valore è presente ed è estratto correttamente*

Altri tipi di liste

- ***Liste circolari***: il puntatore al prossimo elemento della coda della lista punta alla testa della lista stessa
  - Se la lista è anche doppia, il puntatore al precedente elemento della testa della lista punta alla coda della lista stessa
- ***Liste con sentinella:*** Liste che contengono, al posto del puntatore alla testa, un ulteriore ***elemento finto (dummy)***
  - Sempre presente, anche se la lista è vuota

Liste con sentinella

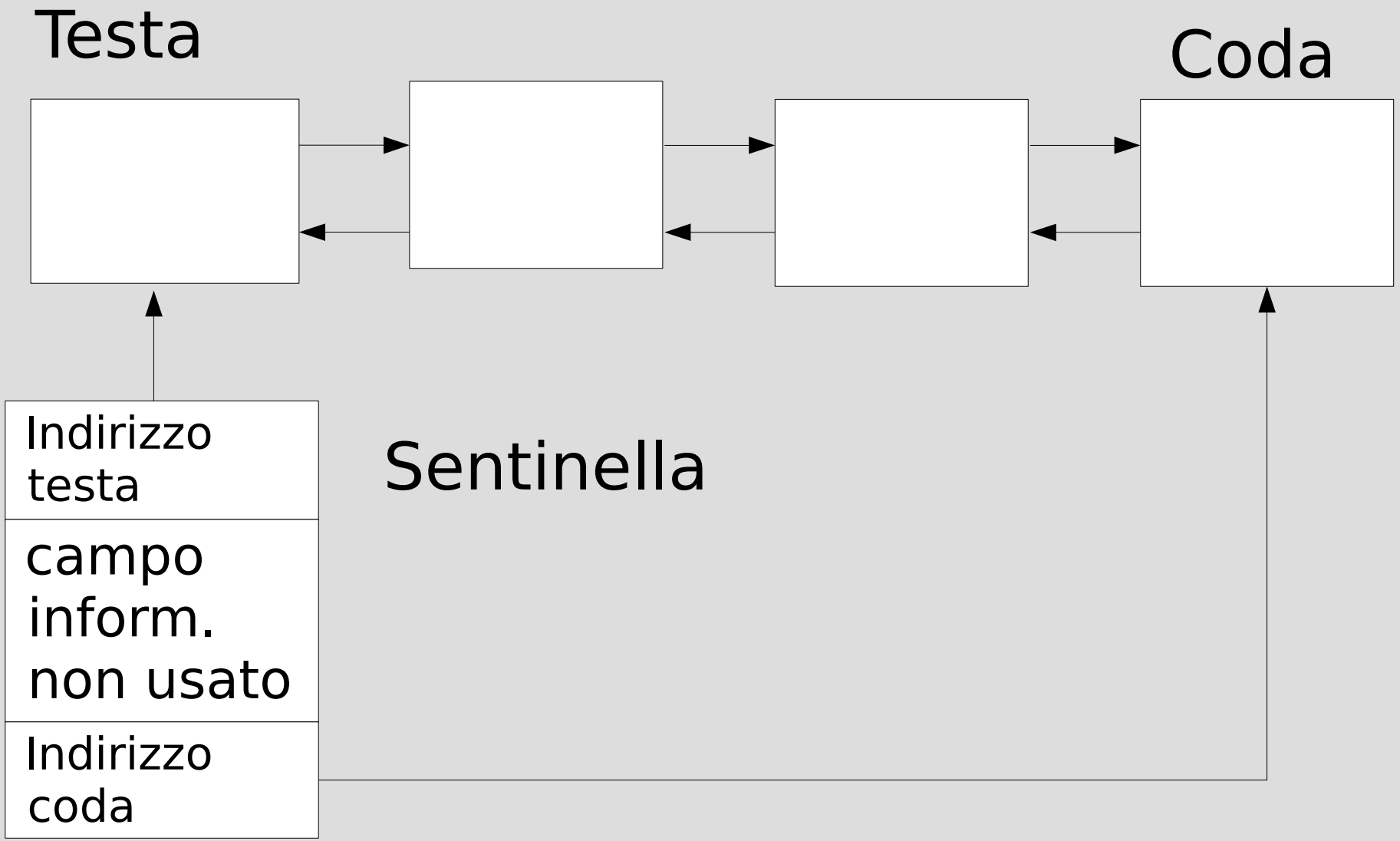
- Una delle cose che ha complicato gli algoritmi sulle liste visti finora è il fatto che il puntatore alla testa della lista era un ***oggetto di tipo diverso*** dagli elementi della lista stessa
- Abbiamo pertanto dovuto ***gestire a parte*** i casi in cui era necessario modificare il puntatore alla testa della lista

  --> Introduzione dell'elemento dummy

Liste con sentinella

- In una ***lista semplice con sentinella***, il campo puntatore dell'elemento sentinella punta alla testa della lista
- In una ***lista doppia con sentinella***
  - il campo *puntatore al prossimo* elemento dell'elemento sentinella punta alla testa della lista
  - il campo *puntatore al precedente* elemento dell'elemento sentinella punta alla coda della lista

Schema


Testa
Coda
Indirizzo testa
campo inform. non usato
Indirizzo coda
Sentinella