Liste 1

Esercitazione 11

25/11/2024 - Alessandro Montenegro

Recap: Memoria Dinamica

Variabili... Finora

Variabili Automatiche

- Dichiarate nelle **funzioni** (o globalmente)
- Salvate automaticamente nello **stack**
- **Ciclo di vita**: da quando la funzione entra in azione fino alla sua terminazione
- **Pro**: facile gestione
- **Contro**: fare attenzione a cosa si restituisce (e.g., non possiamo restituire formalmente array)

Variabili... Finora

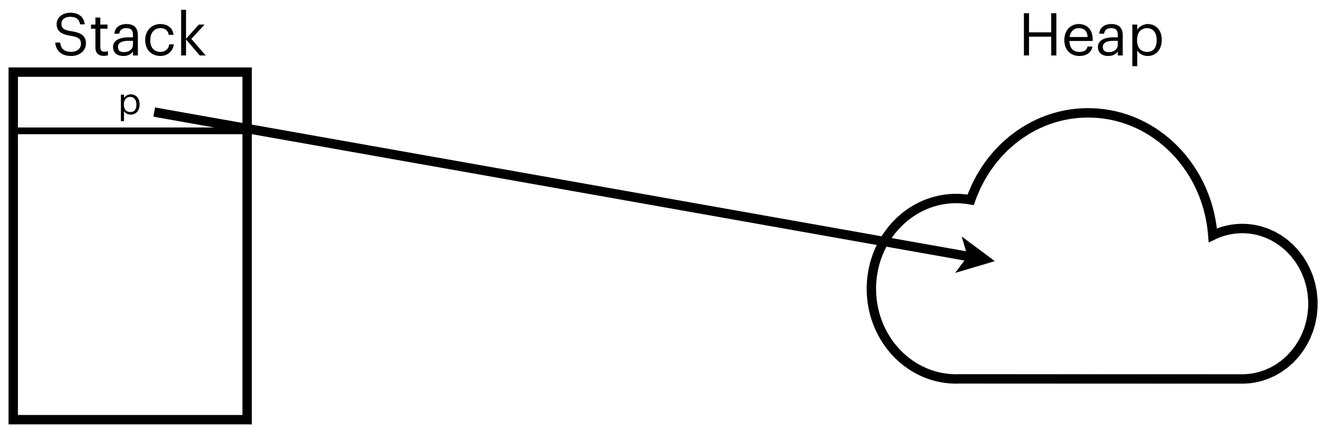
Variabili Automatiche e Array

- Array vanno **sovradimensionati** (spreco di spazio)
- **NON** si possono **dimensionare** con **variabili**, VLA
- **NON** bisogna **superare** la loro **dimensione massima**
- Bisogna tenere traccia delle loro **dimensioni effettive**
- Non possiamo **gestire senza sprechi** i programmi in cui il **numero di dati** è noto a **run-time**

Memoria Dinamica

Come e Quando

- Per ovviare a tutti i problemi di prima, possiamo memorizzare i dati in un’altra area di memoria, chiamata **Heap**
- Accessibile solo tramite **puntatori** (che rimangono variabili automatiche nello stack)
- **Referenziabile da ogni ambiente** (non viene sovrascritta in automatico)



Memoria Dinamica

Allocazione

```
#include <stdlib.h>

void * malloc(int);

tipo* p = (tipo*) malloc(sizeof(tipo));
```

Se fallisce (spazio esaurito), restituisce NULL

La **variabile puntatore p** (automatica, quindi salvata nello stack) memorizza l'**indirizzo di memoria nell'Heap** dove è allocata una **porzione di spazio** della **dimensione "tipo"**

Memoria Dinamica

Deallocazione

```
#include <stdlib.h>

void free(void*);

free(p);
```

Viene **liberato** lo **spazio nell'Heap** a cui punta p

p sarà **ancora utilizzabile** (variabile automatica)

Se per qualche ragione **perdiamo l'indirizzo in p**, quello **spazio** nell'Heap **non è più raggiungibile** e si accumula "garbage"

Memoria Dinamica

Pro e Contro

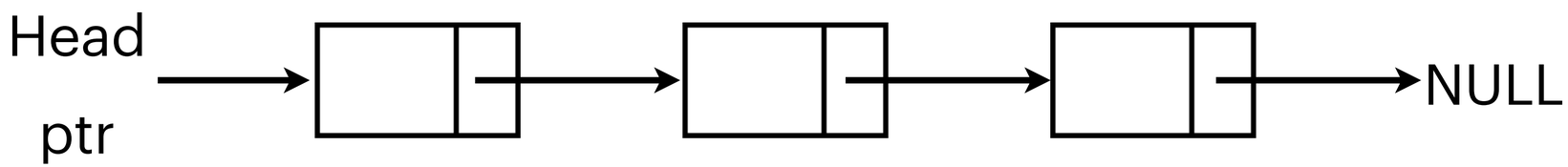
- **Pro**: risolve i problemi delle variabili automatiche prima citati
- **Contro**: la gestione deve essere accorta
  - Si deve **liberare lo spazio** quando non serve più
  - Si deve fare attenzione all'**uso dei puntatori** (e.g., non accedere alla memoria quando l'indirizzo memorizzato è NULL)
  - **Nota**: errore tipico da terminale per l'errata gestione della memoria è `segmentation fault`

Recap: Liste

Liste

Strutture Ricorsive

```c
typedef struct NodeStruct {
    int data;
    struct NodeStruct * next;
} Node;
typedef Node * pNode;
```



Liste

Operazioni Principali

- **Aggiunta**
  - Aggiunta di un Elemento in Coda
  - Aggiunta di un Elemento in Testa
  - Aggiunta di un Elemento in una Posizione Specifica
  - Aggiunta Ordinata senza Ripetizioni
- **Ricerca e Analisi**
  - Lunghezza, Massimo/Minimo, Media, etc...
- **Eliminazione**
  - Eliminazione di tutta la Lista
  - Eliminazione di un Elemento in una posizione specifica
  - Eliminazione di tutti gli Elementi che rispettano una certa condizione

Aggiunta

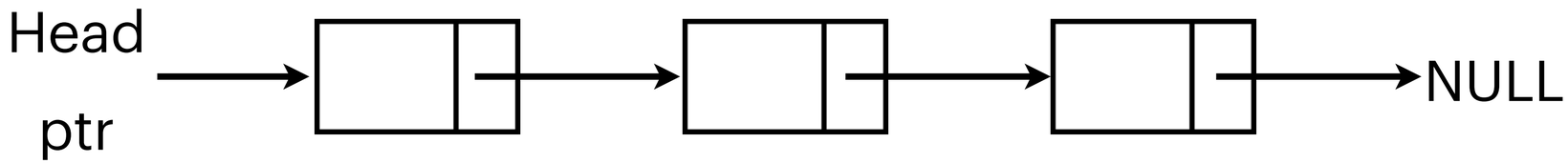
Aggiunta in Coda

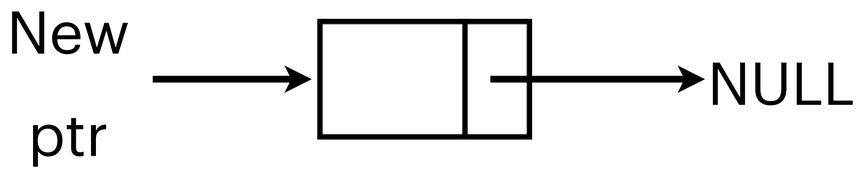
Strategia

- **Creo** il nuovo nodo (malloc + assegno i campi del nodo)
- **Scorro** la lista fino alla fine
- **Cambio** il ptr a **next** dell'**ultimo** nodo con l'indirizzo del nuovo nodo puntato

Aggiunta in Coda

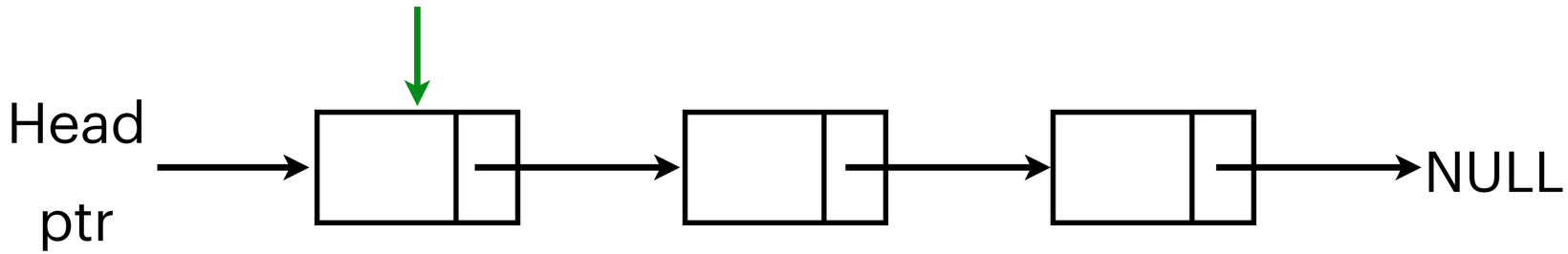
Esempio

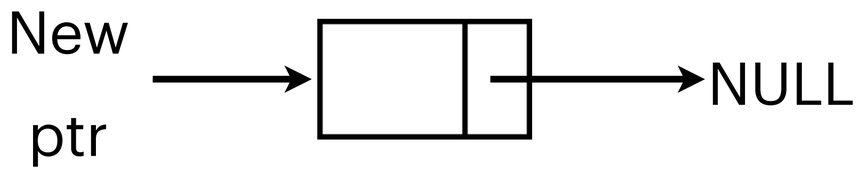




Aggiunta in Coda

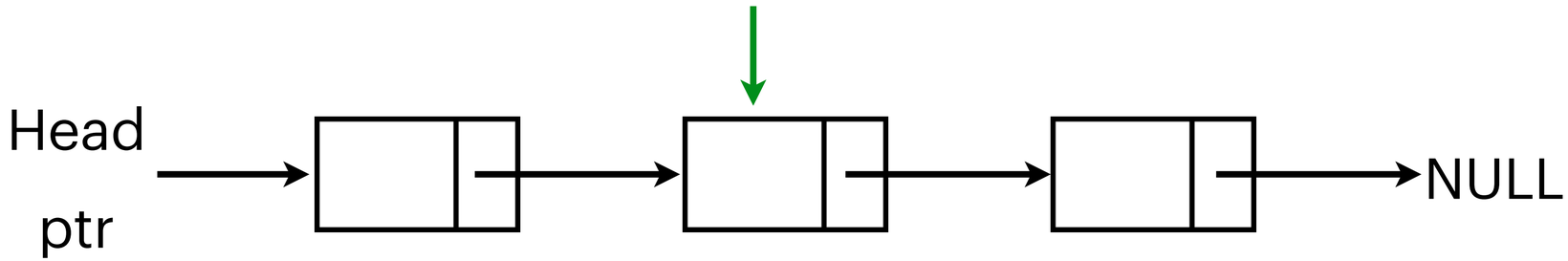
Esempio

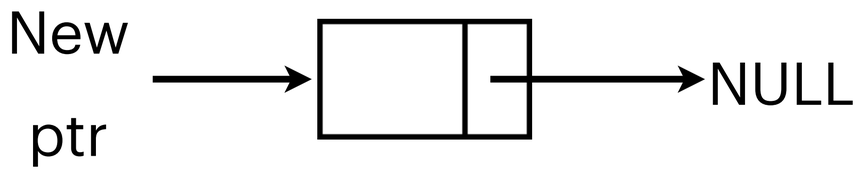




Aggiunta in Coda

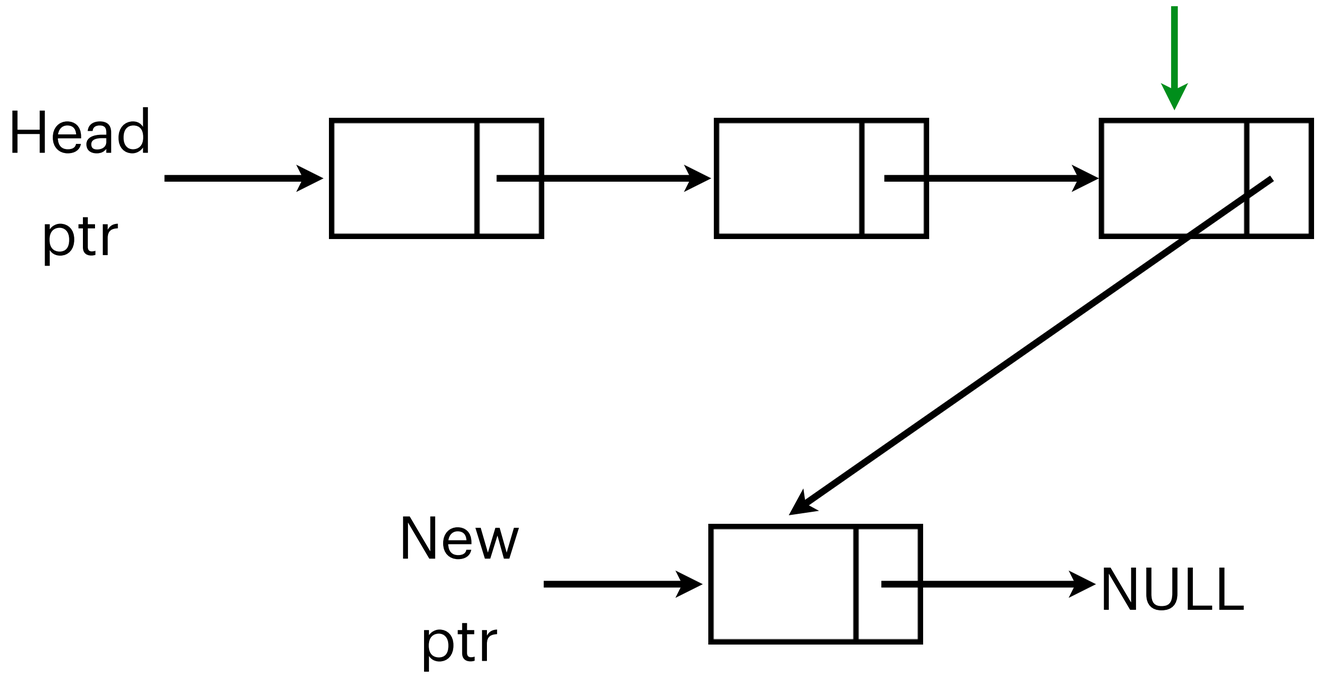
Esempio





Aggiunta in Coda

Esempio



Aggiunta in Coda

Come?

- Uso un puntatore “curr” inizializzato a “head”
- Scorro aggiornando “curr = curr->next”
- Quando “curr -> next == NULL” cambio “curr -> next = new_ptr”

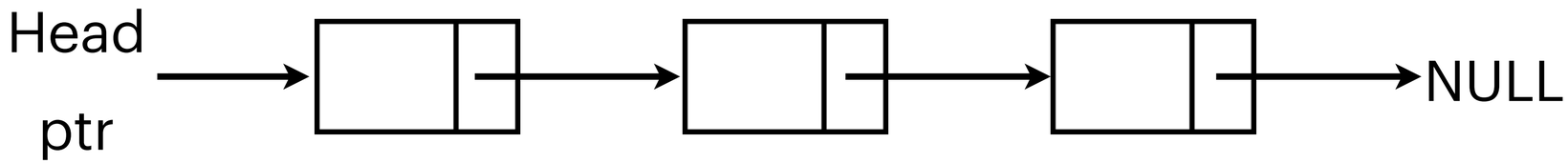
Aggiunta in Testa

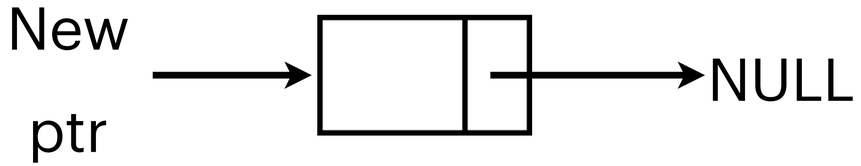
Strategia

- **Creo** il nuovo nodo (malloc + assegno i campi del nodo)
- Faccio puntare il **next del nuovo** nodo alla **testa** della lista
- **Restituisco** la **nuova testa** della lista, che sarà il puntatore al nuovo nodo

Aggiunta in Testa

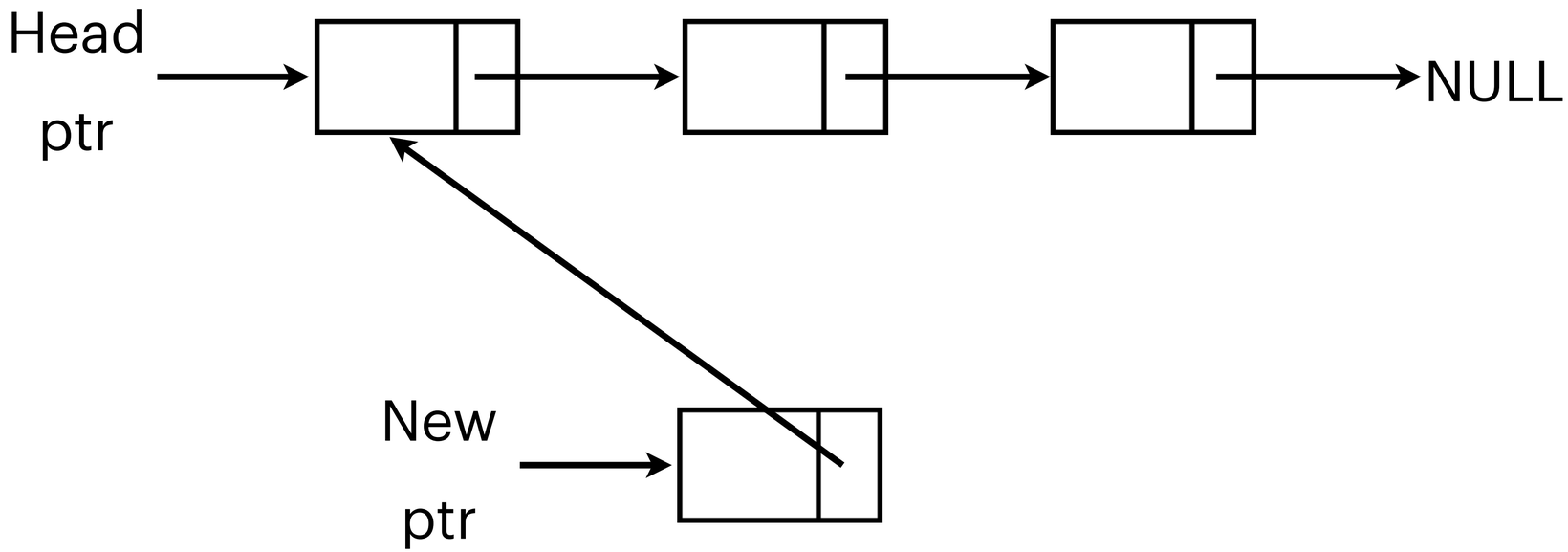
Esempio





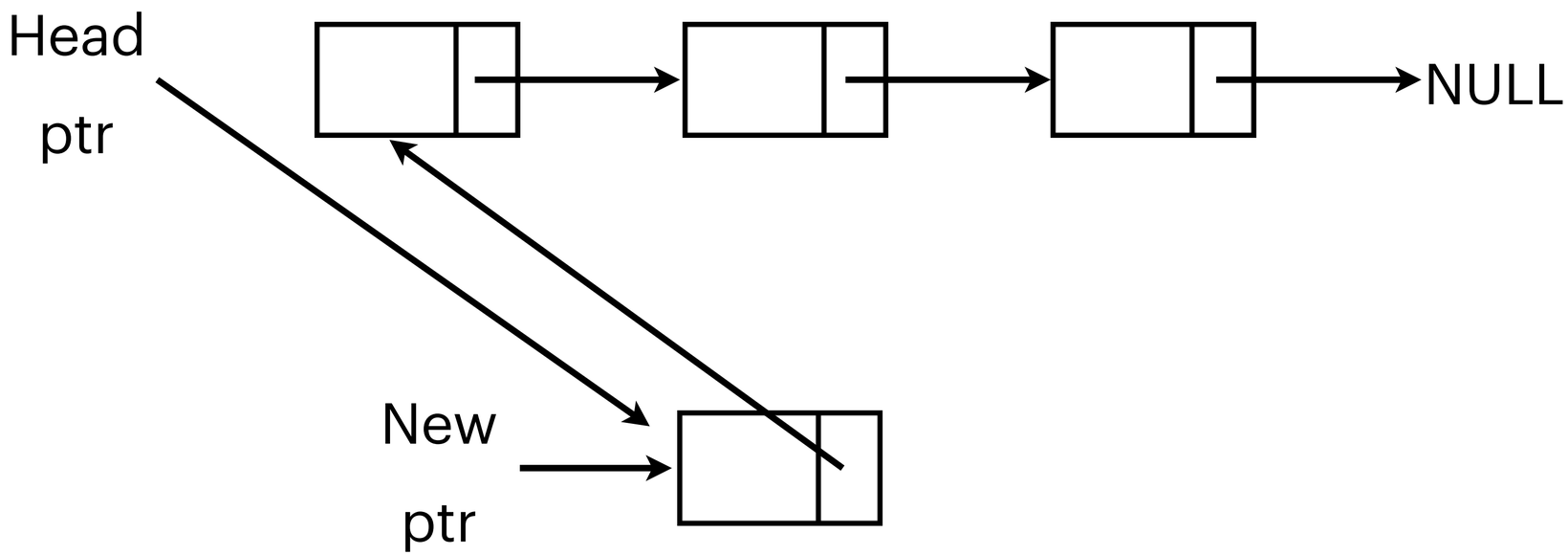
Aggiunta in Testa

Esempio



Aggiunta in Testa

Esempio



Aggiunta in una Posizione Specifica

Strategia

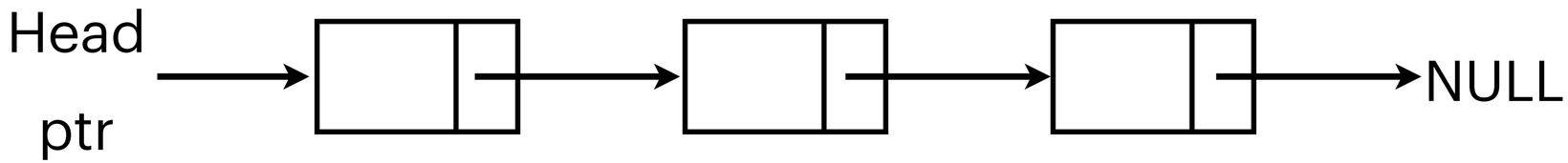
- **Creo** il nuovo nodo (malloc + assegno i campi del nodo)
- Controllo se la **posizione è la testa** della lista, in tal caso faccio un inserimento in Testa
- **Scorro** fino alla **posizione** desiderata (se valida)
- Inserisco il **nodo nella posizione desiderata**, facendo scorrere la lista

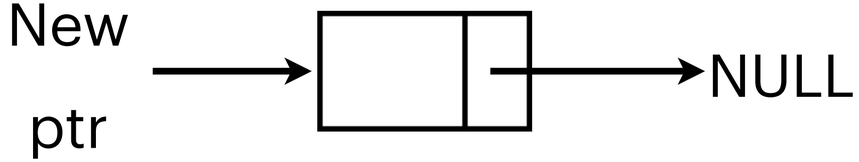
Aggiunta in una Posizione Specifica

Esempio

Inserimento in posizione

1



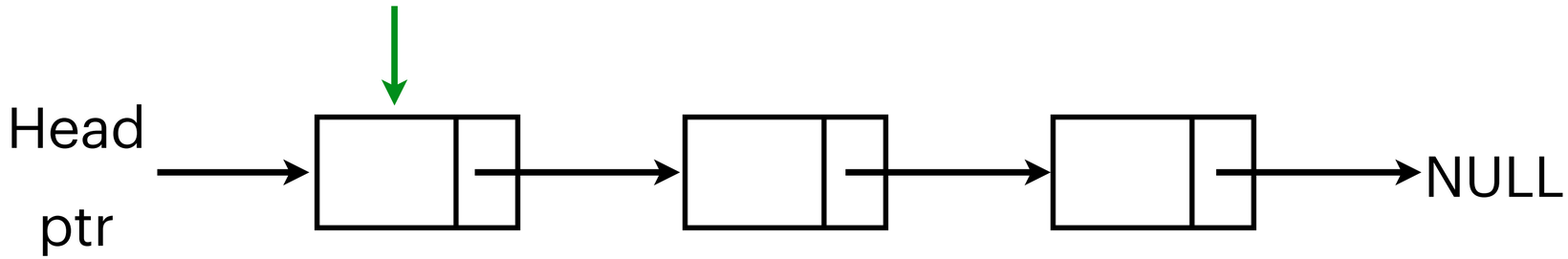


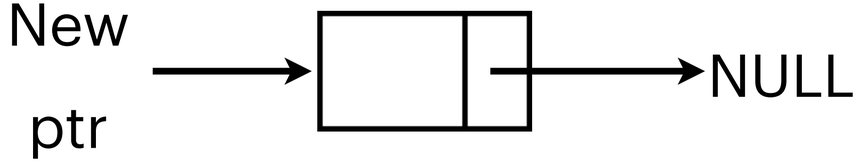
Aggiunta in una Posizione Specifica

Esempio

Inserimento in posizione

1



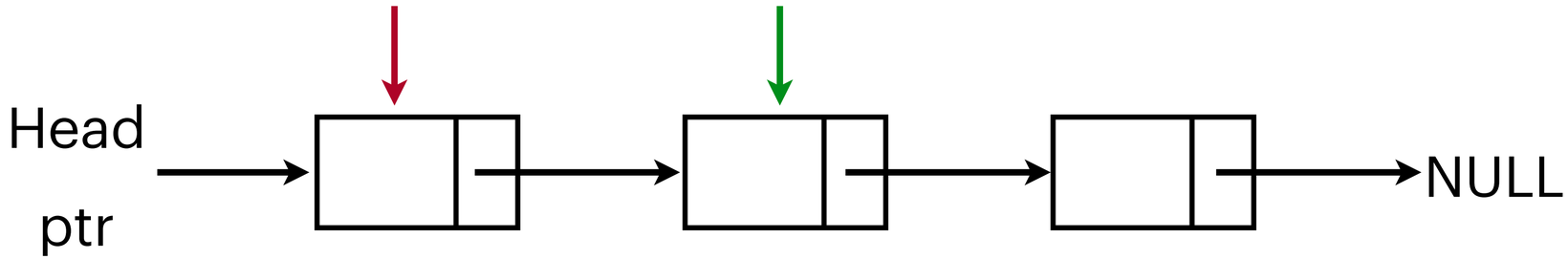


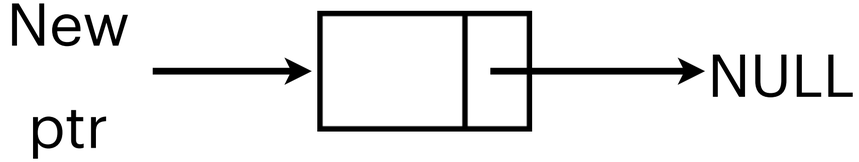
Aggiunta in una Posizione Specifica

Esempio

Inserimento in posizione

1



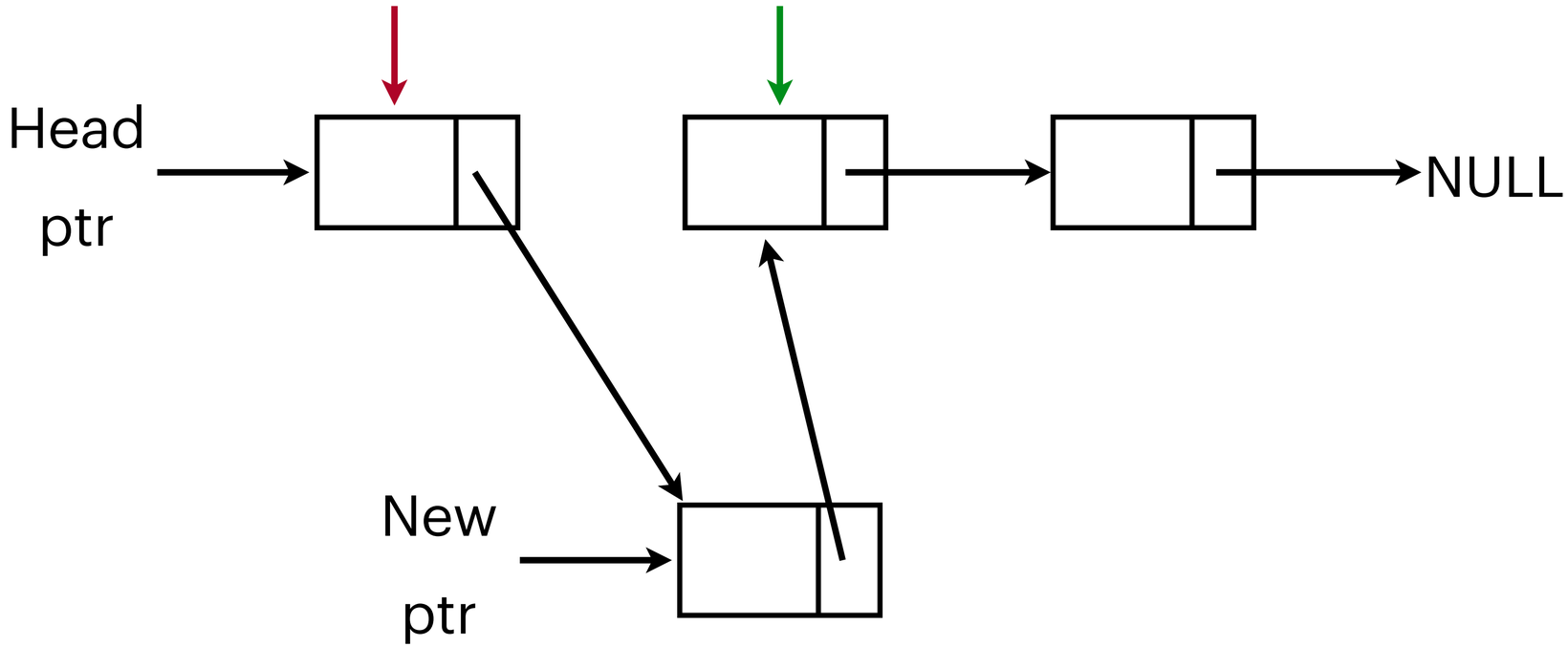


Aggiunta in una Posizione Specifica

Esempio

Inserimento in posizione

1



Aggiunta in una Posizione Specifica

Come?

- Uso un **puntatore** “curr” inizializzato a “head”, e un **puntatore** “prec” inizializzato a NULL
- Uso una **variabile** “index” che tiene conto **dell’indice**
- **Scorro aggiornando** “prec = curr”, “curr = curr->next” e “index++”
- Quando “index == posizione desiderata” cambio “prec -> next = new_ptr” e “new_ptr->next = curr”
- Se l’indice **non è valido** (troppo grande) il nuovo nodo va **DEALLOCATO**

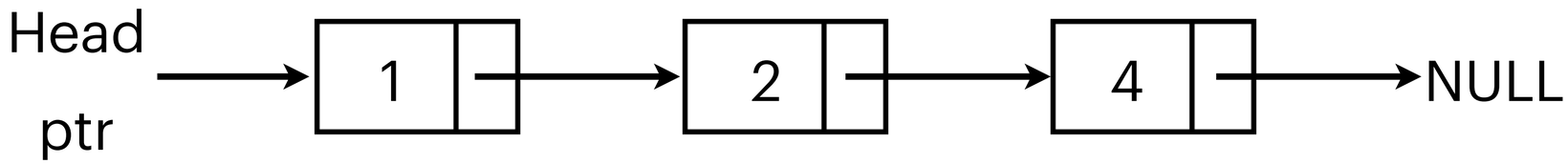
Aggiunta Ordinata (Senza Ripetizioni)

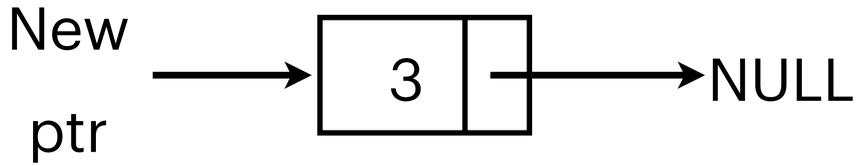
Strategia

- **Creo** il nuovo nodo (malloc + assegno i campi del nodo)
- Controllo se devo aggiungere **prima della testa** della lista, in tal caso faccio un inserimento in Testa
- **Scorro** fino alla a quando gli elementi sono > del nuovo nodo desiderata (se valida)
- Se il valore è già presente, non inserisco (e dealloco)
- Inserisco il **nodo nella posizione desiderata**, facendo scorrere la lista

Aggiunta Ordinata (Senza Ripetizioni)

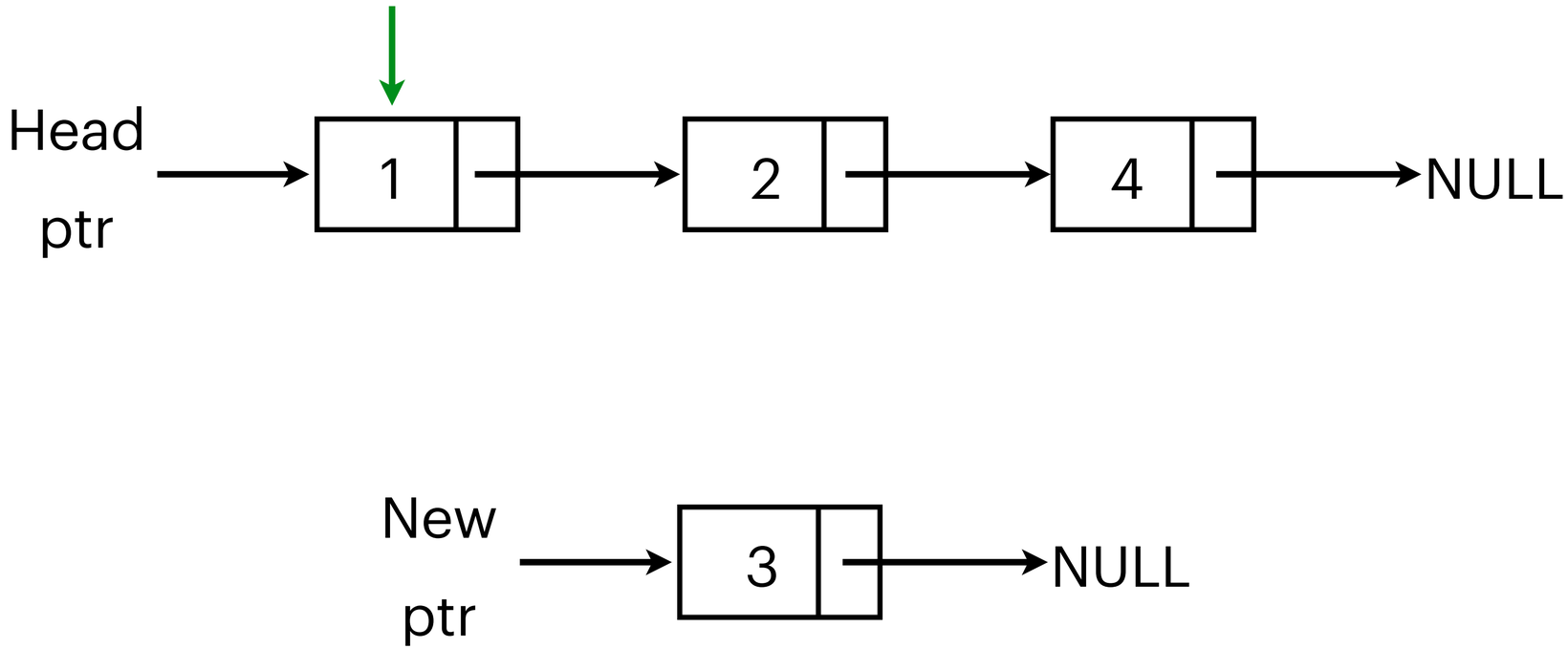
Esempio





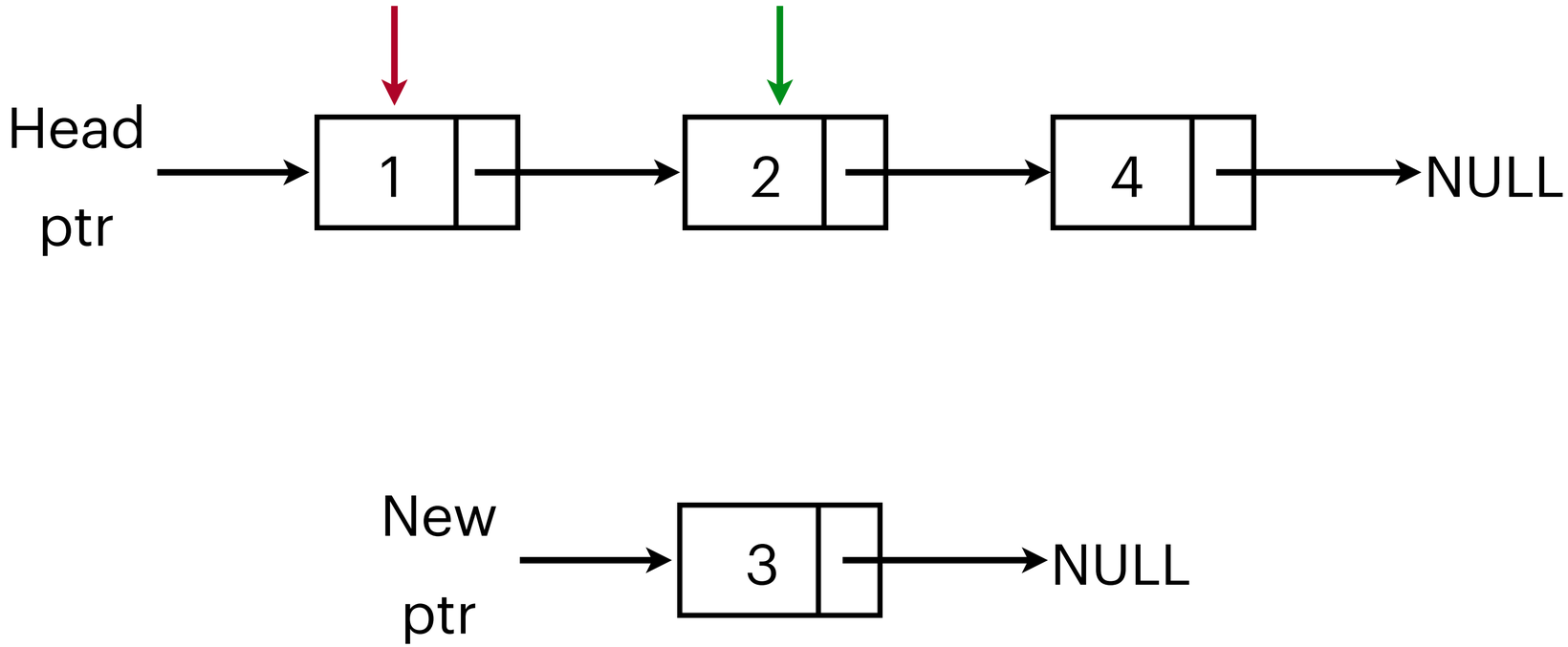
Aggiunta Ordinata (Senza Ripetizioni)

Esempio



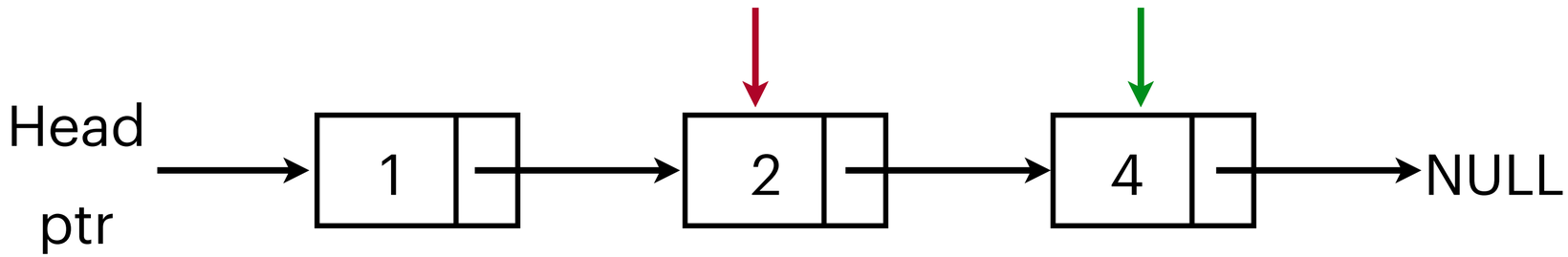
Aggiunta Ordinata (Senza Ripetizioni)

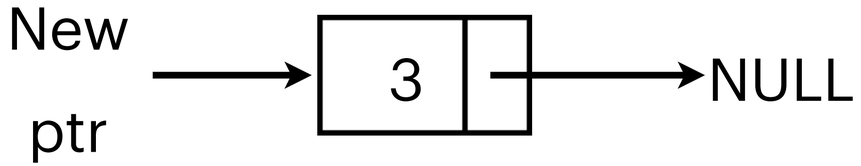
Esempio



Aggiunta Ordinata (Senza Ripetizioni)

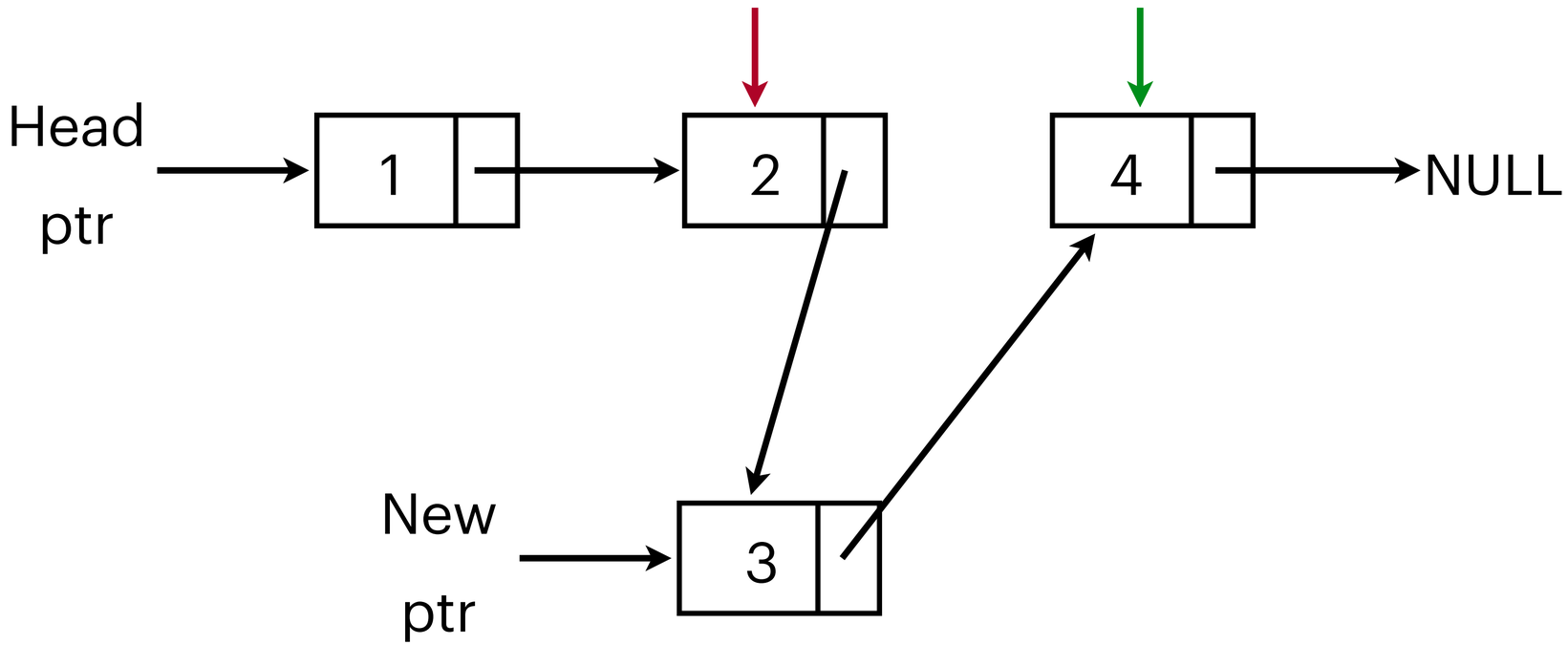
Esempio





Aggiunta Ordinata (Senza Ripetizioni)

Esempio



Aggiunta in una Posizione Specifica

Come?

- Uso un **puntatore** “curr” inizializzato a “head”, e un **puntatore** “prec” inizializzato a NULL
- **Scorro aggiornando** “prec = curr” e “curr = curr->next”
- Quando “curr->valore > new_ptr->valore” cambio “prec -> next = new_ptr” e “new_ptr->next = curr”
- Se la **lista è vuota**, restituisco new_ptr
- Se la testa ha una valore > di new_ptr->valore, **inserisco in Testa**
- Se l’**elemento è già presente** il nuovo nodo va **DEALLOCATO**

Ricerca e Analisi

Ricerca e Analisi

Strategie

- Può essere richiesto di analizzare una lista già allocata
- **Ad esempio**: trovare il massimo valore, trovare la media, contare gli elementi, effettuare una copia di un numero particolare di elementi, etc...
- Bisogna usare i meccanismi visti finora: scorro la lista e analizzo i valori dei nodi

Eliminazione

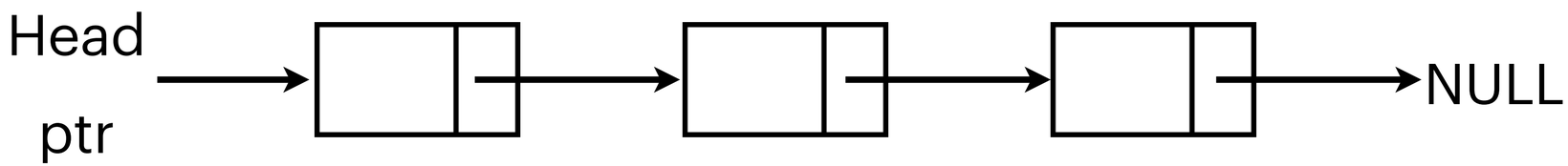
Eliminazione di Tutta la Lista

Strategia

- **Scorro** la lista
- **Salvo** in un nuovo puntatore l’**elemento corrente**
- **Aggiorno la testa** della lista
- **Dealloco** il nodo puntato

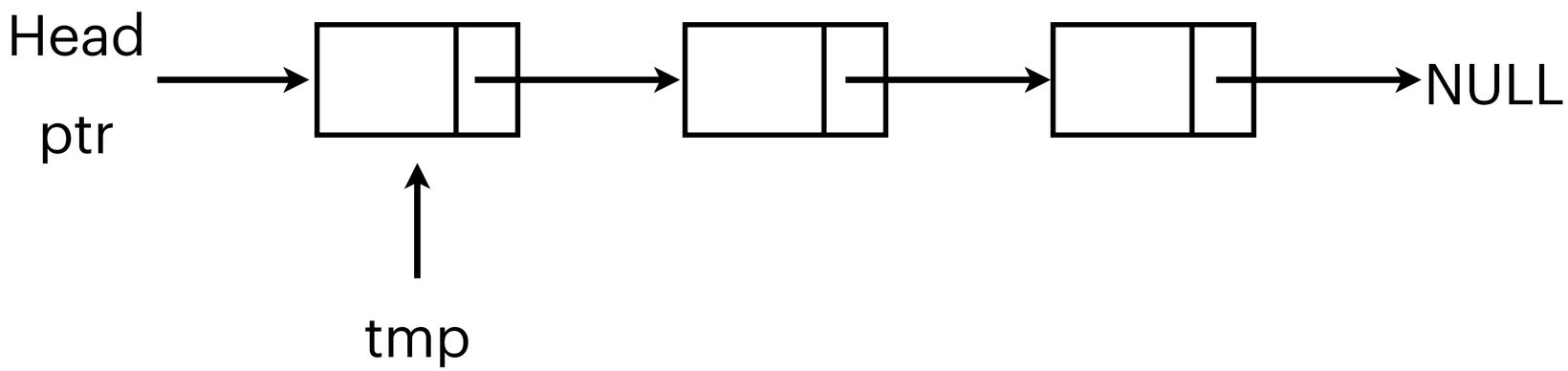
Eliminazione di Tutta la Lista

Esempio



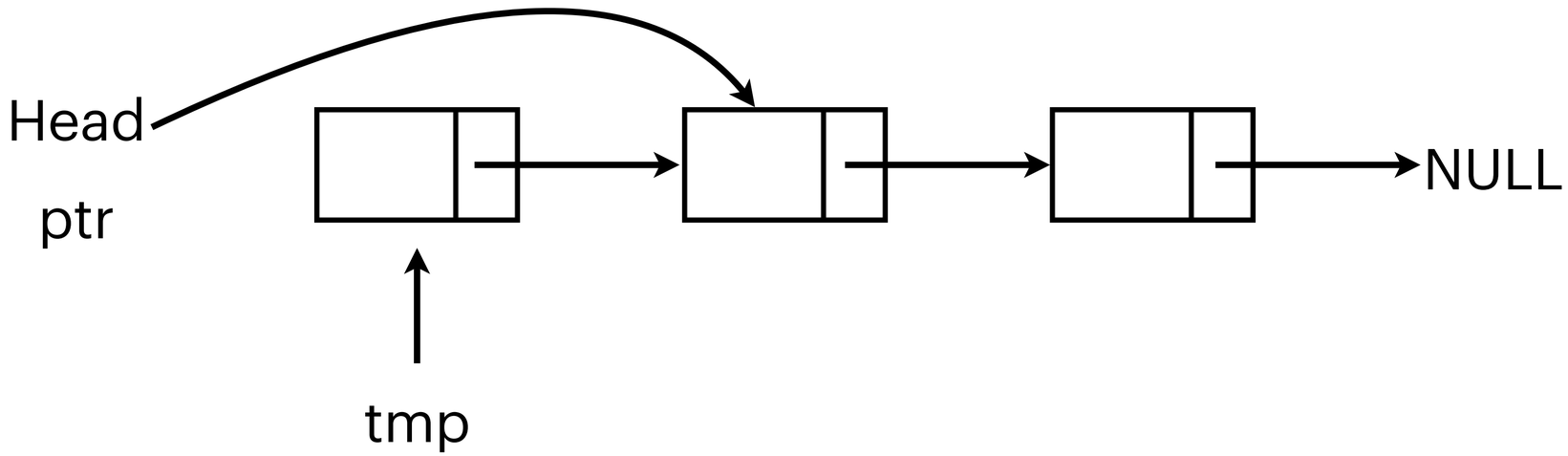
Eliminazione di Tutta la Lista

Esempio



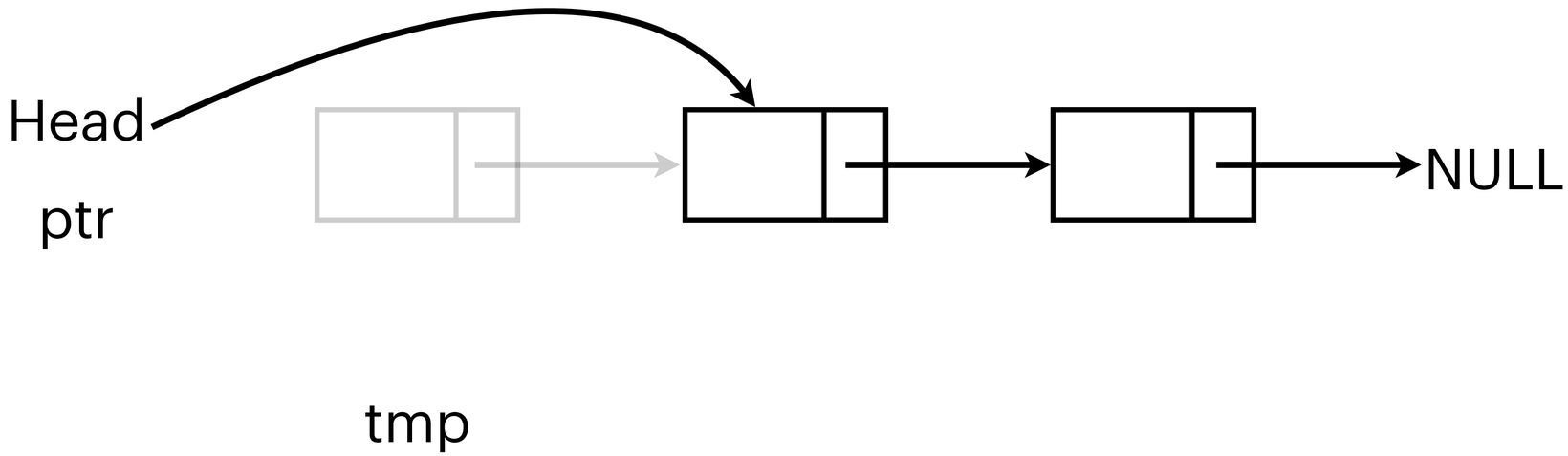
Eliminazione di Tutta la Lista

Esempio



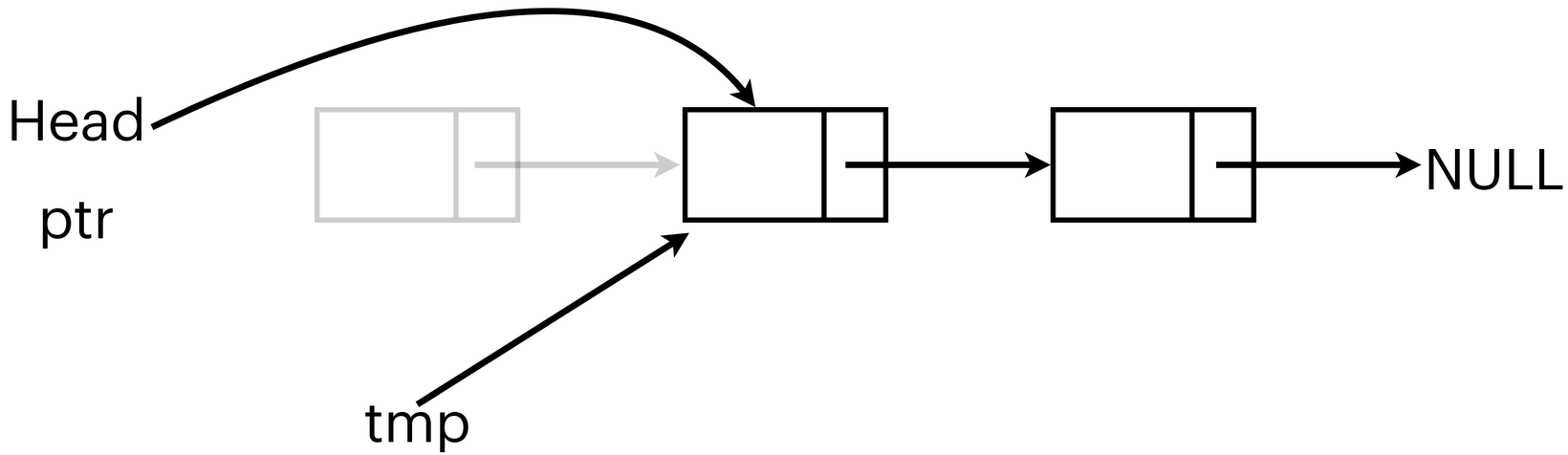
Eliminazione di Tutta la Lista

Esempio



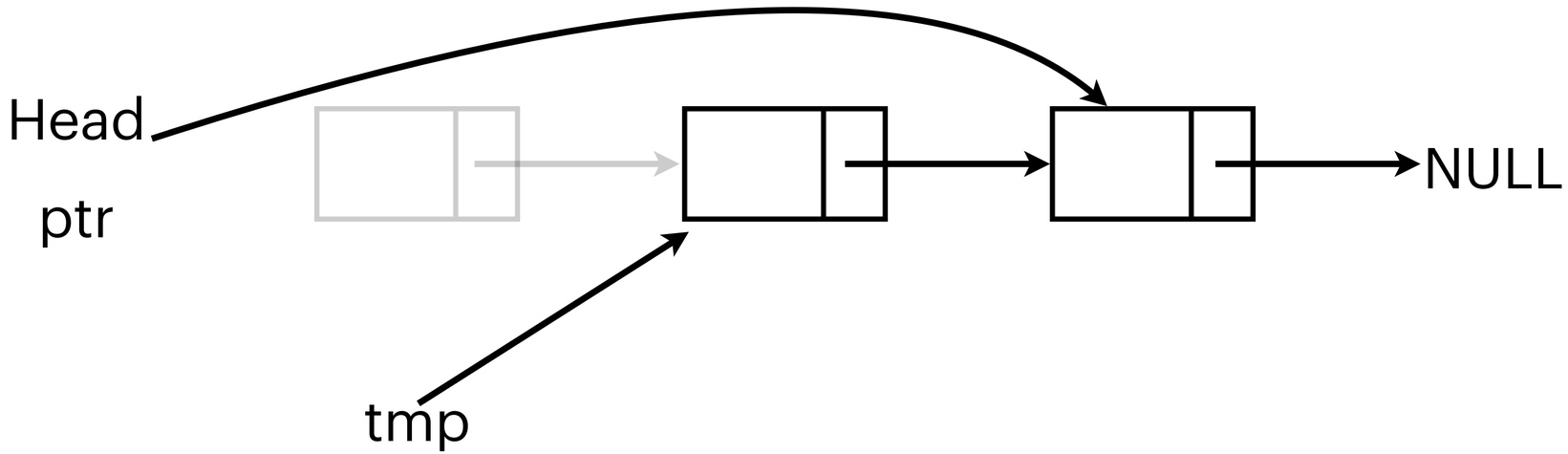
Eliminazione di Tutta la Lista

Esempio



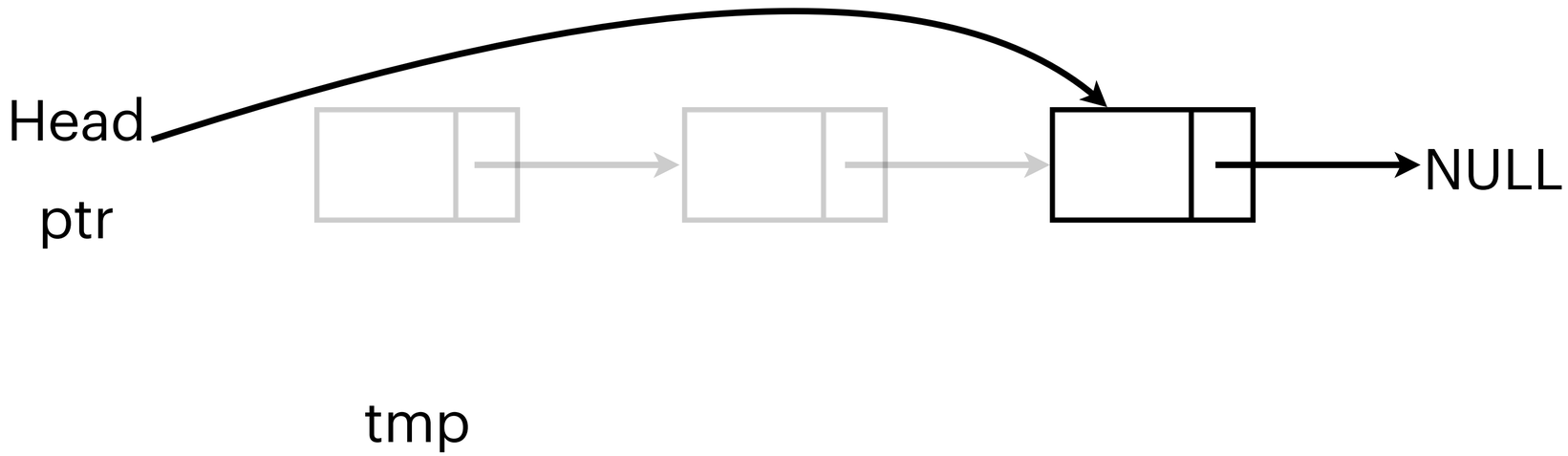
Eliminazione di Tutta la Lista

Esempio



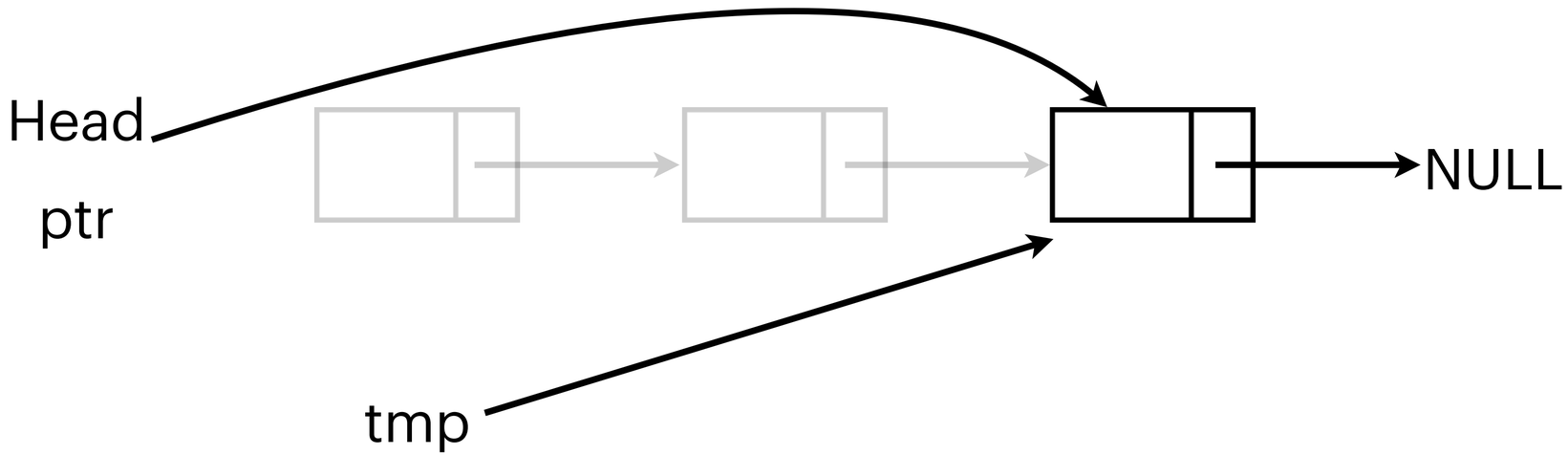
Eliminazione di Tutta la Lista

Esempio



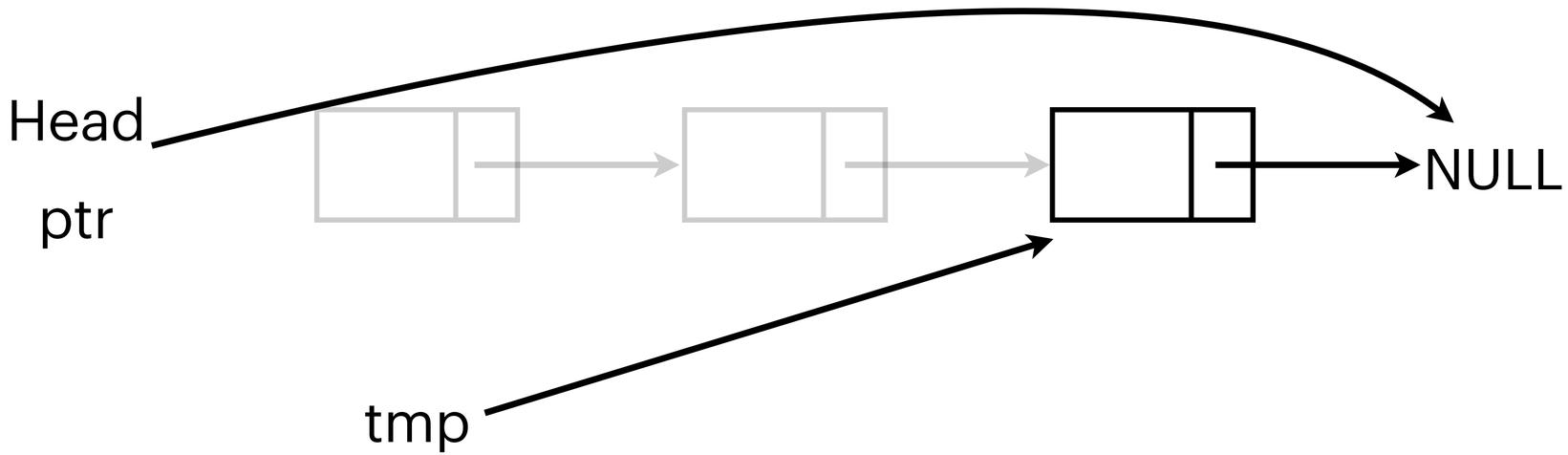
Eliminazione di Tutta la Lista

Esempio



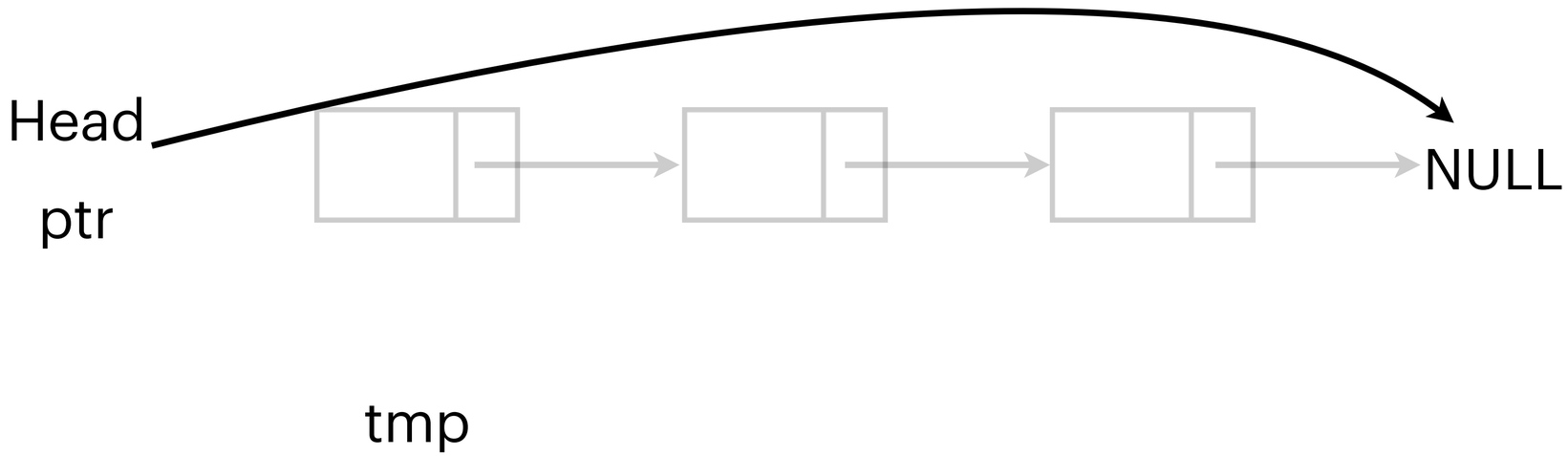
Eliminazione di Tutta la Lista

Esempio



Eliminazione di Tutta la Lista

Esempio



Eliminazione di un Elemento in Posizione i

Strategia

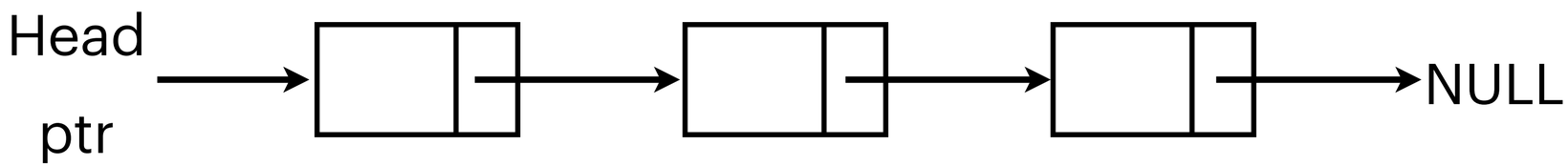
- Controllo se devo eliminare la **Testa**
- **Scorro** la lista fino alla posizione prevista
- **Dealloco** il nodo facendo **puntare il suo predecessore al suo successore**

Eliminazione di un Elemento in Posizione i

Esempio

Elimino in posizione

1

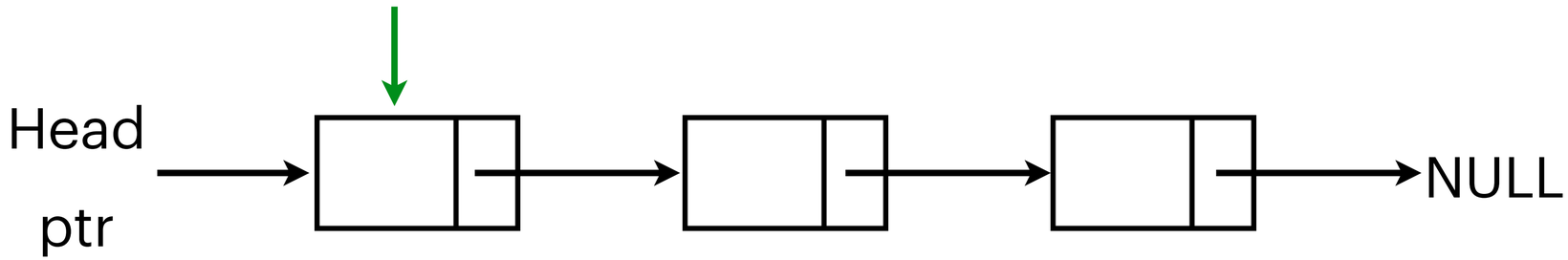


Eliminazione di un Elemento in Posizione i

Esempio

Elimino in posizione

1

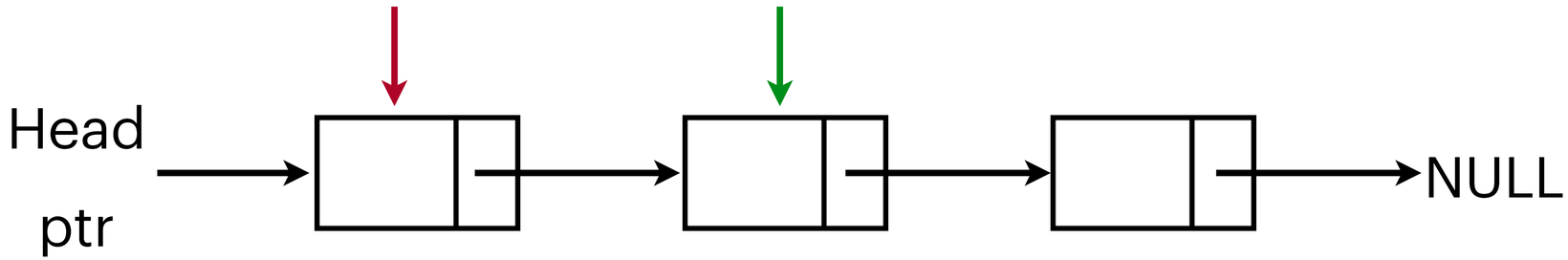


Eliminazione di un Elemento in Posizione i

Esempio

Elimino in posizione

1

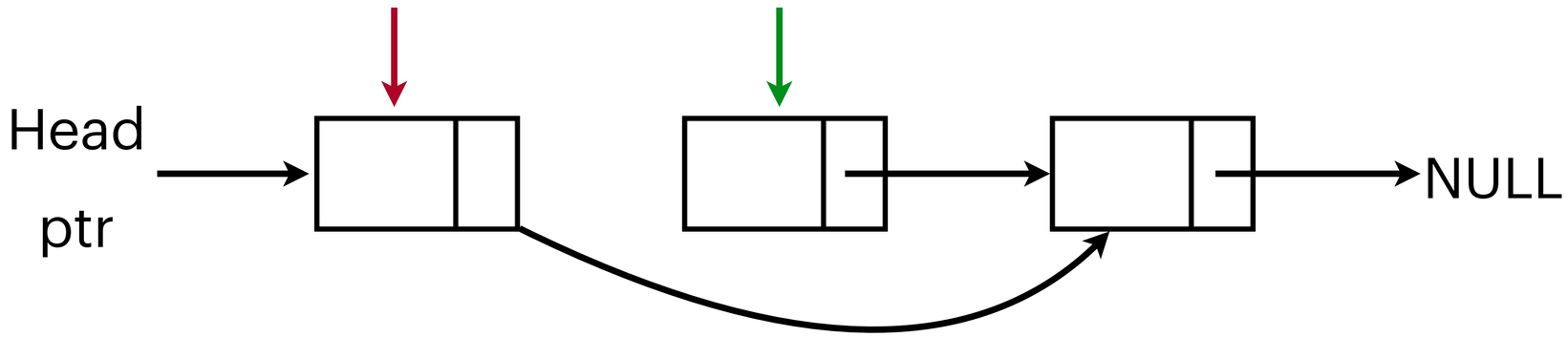


Eliminazione di un Elemento in Posizione i

Esempio

Elimino in posizione

1

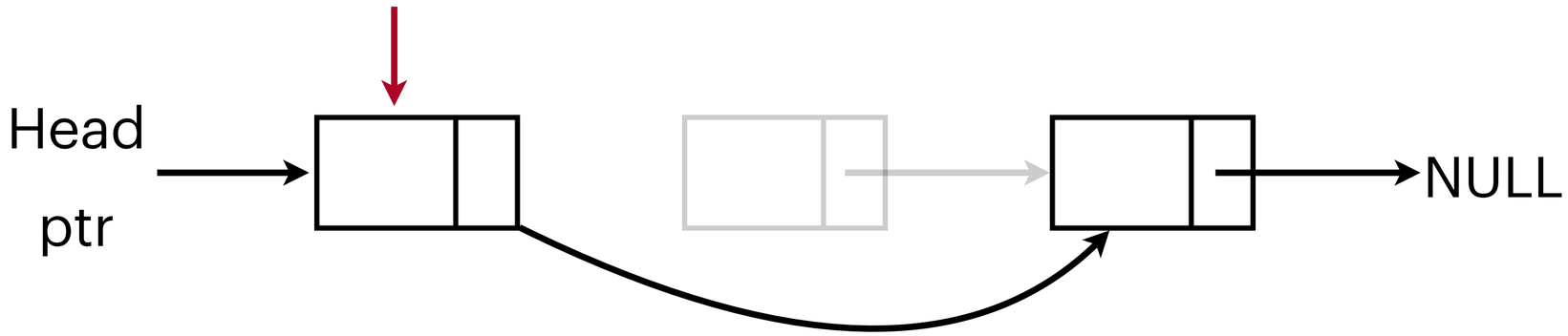


Eliminazione di un Elemento in Posizione i

Esempio

Elimino in posizione

1



Eliminazione di un Elemento in Posizione i

Come?

- Uso un **puntatore** “curr” inizializzato a “head”, e un **puntatore** “prec” inizializzato a NULL
- Uso una **variabile** “index” che tiene conto **dell’indice**
- **Scorro aggiornando** “prec = curr”, “curr = curr->next” e “index++”
- Quando “index == posizione desiderata” cambio “prec -> next = curr->next” e “new_ptr->next = curr”
- Se l’indice **non è valido** (troppo grande, curr == NULL) il nuovo nodo va **DEALLOCATO**
- Bisogna controllare se bisogna eliminare in testa

Eliminazione di Tutti gli Elementi che Rispettano una Condizione

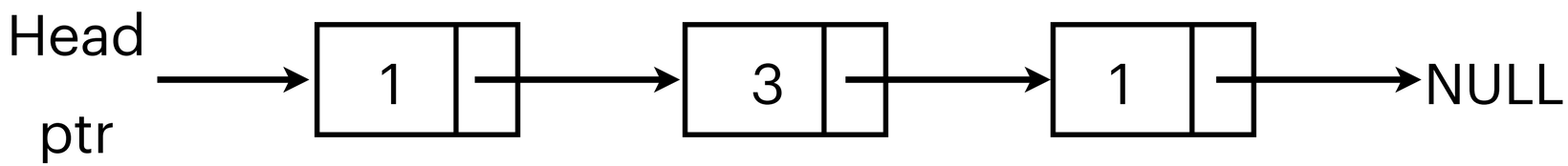
Strategia

- **Scorro** la lista
- Se l’elemento rispetta la condizione, lo elimino (come visto in precedenza)
- Bisogna controllare all’inizio fino a quando devo eliminare elementi in testa

Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

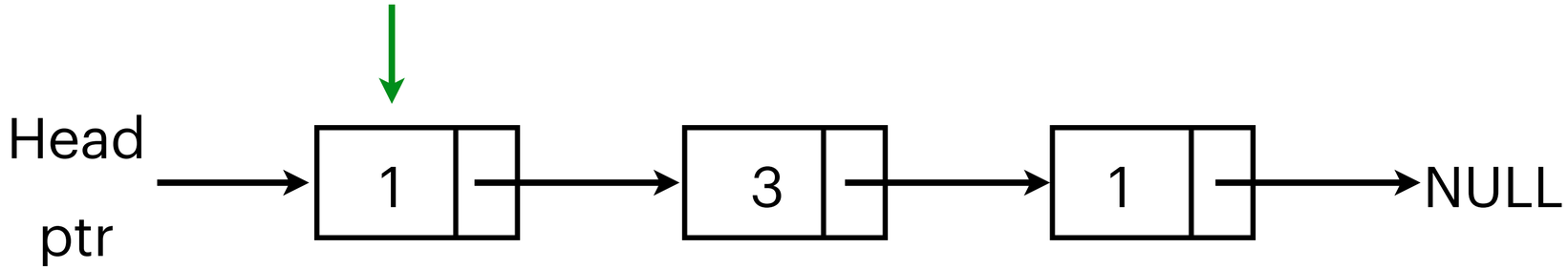
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

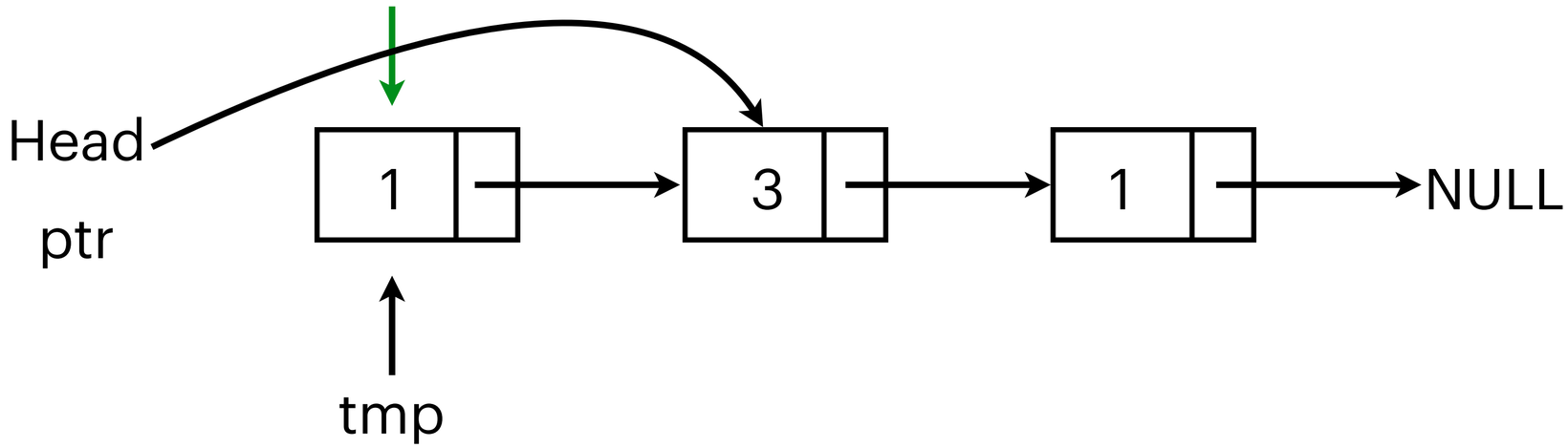
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

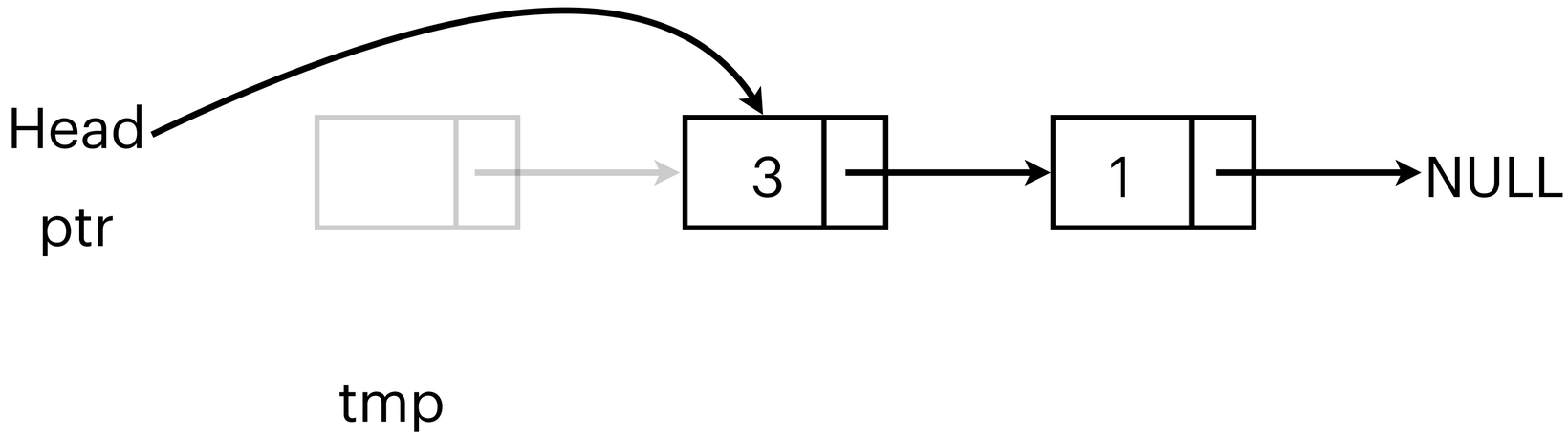
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

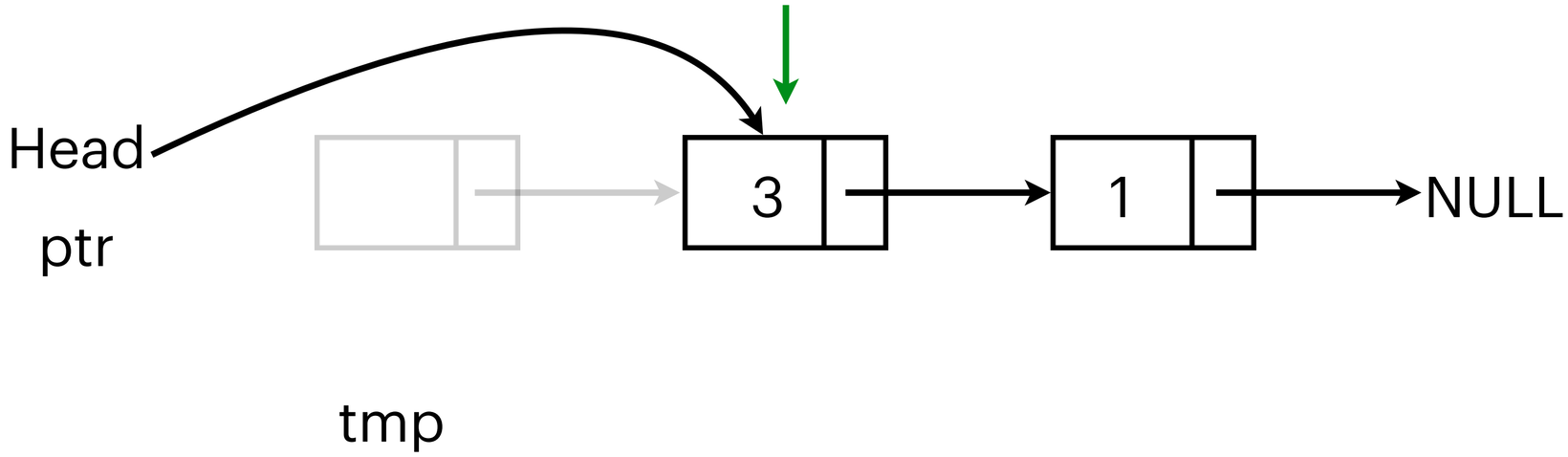
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

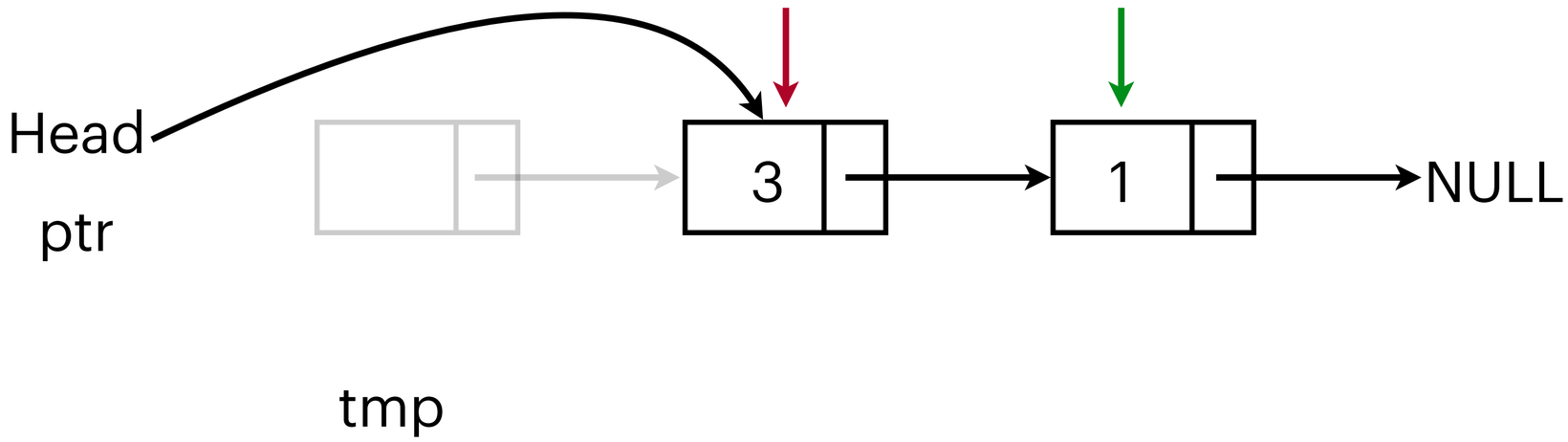
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

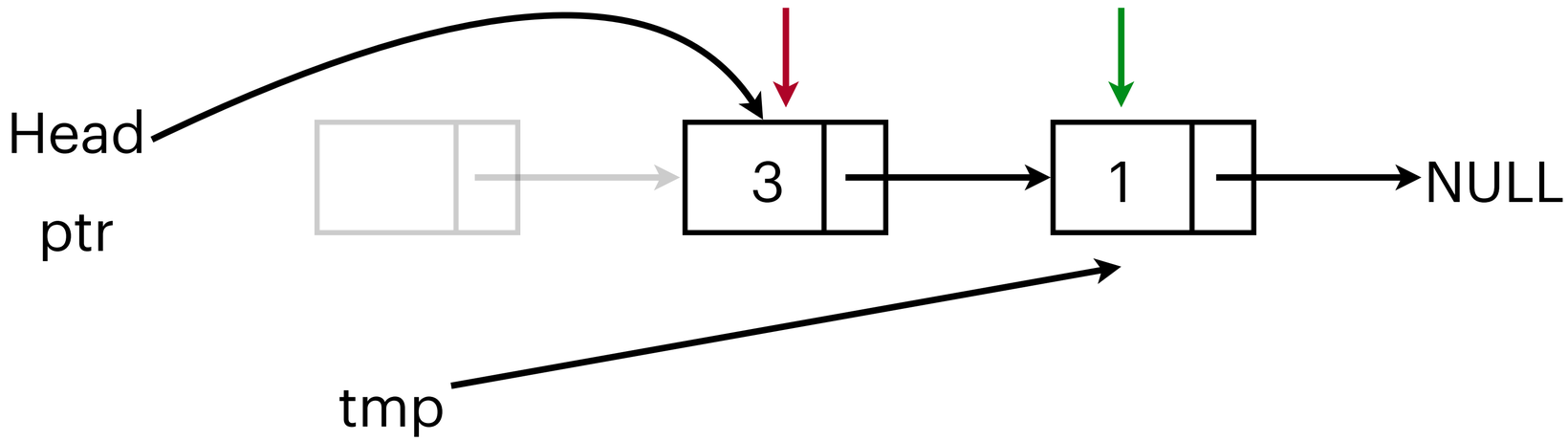
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

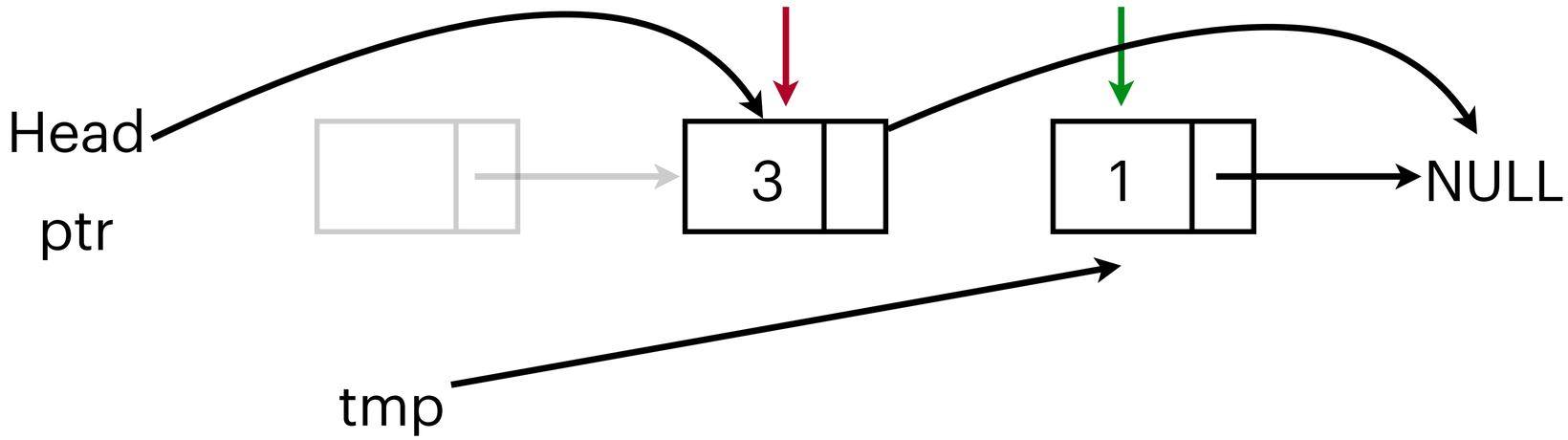
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

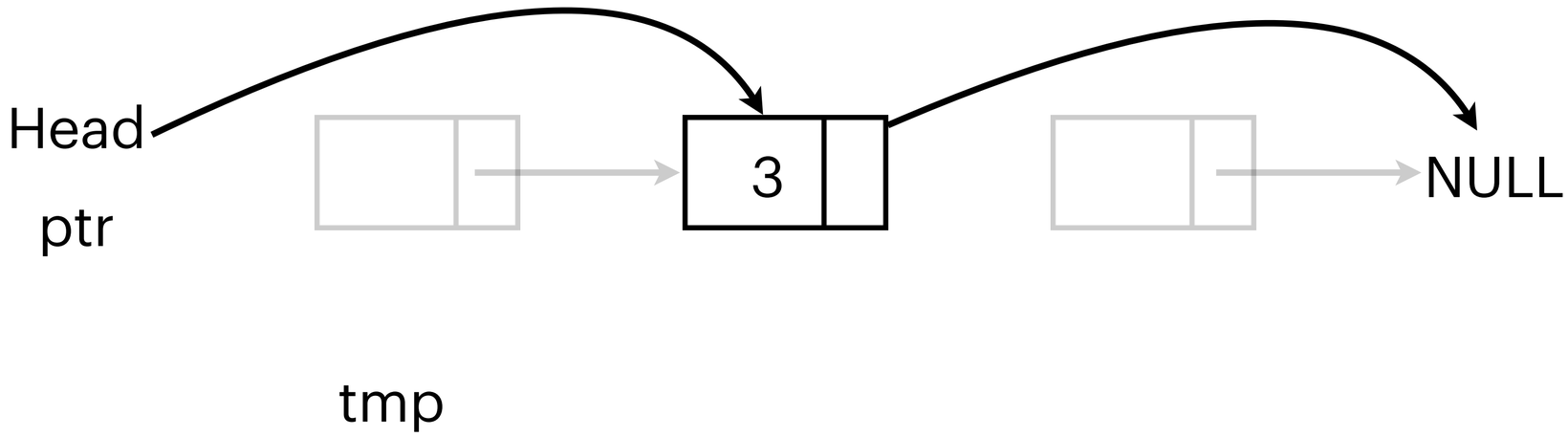
Elimino tutti gli elementi < 2



Eliminazione di Tutti gli Elementi che Rispettano una Condizione

Esempio

Elimino tutti gli elementi < 2



Contatti

Alessandro Montenegro

Mail: alessandro.montenegro@polimi.it

Sito: https://montenegroalessandro.github.io/InfoA2425/index.html