Modulo: Strutture dati dinamiche lineari [P2_08]

Unità didattica: Strutture dati dinamiche lineari (2) [4-T]

Titolo: Principali strutture dinamiche lineari: lista (linked list)

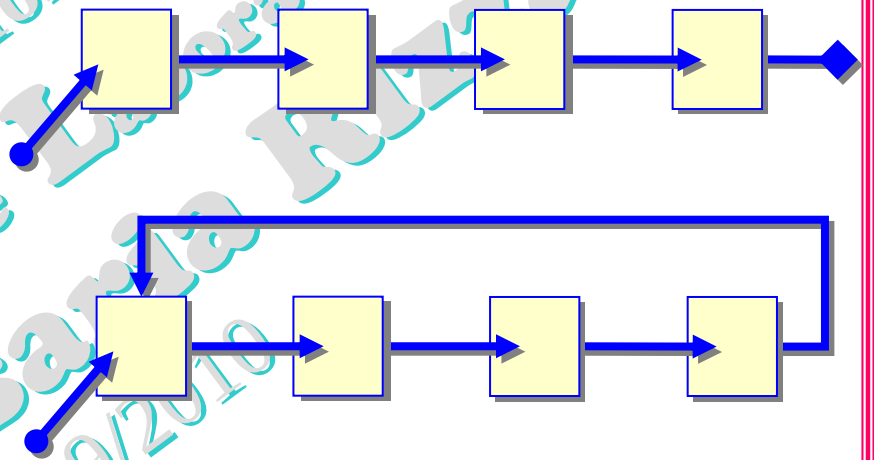
Argomenti trattati:

- ✓ Struttura sequenziale: la lista lineare (linked list)
- ✓ Visita di una lista
- ✓ Inserimento ed eliminazione di un nodo dalla lista
- ✓ Simulazione su array di una lista lineare

Prerequisiti richiesti: fondamenti della programmazione C, generalità sulle strutture dati lineari, array

Tipo di dato astratto (Abstract Data Type): **Lista Lineare (Linked List)**

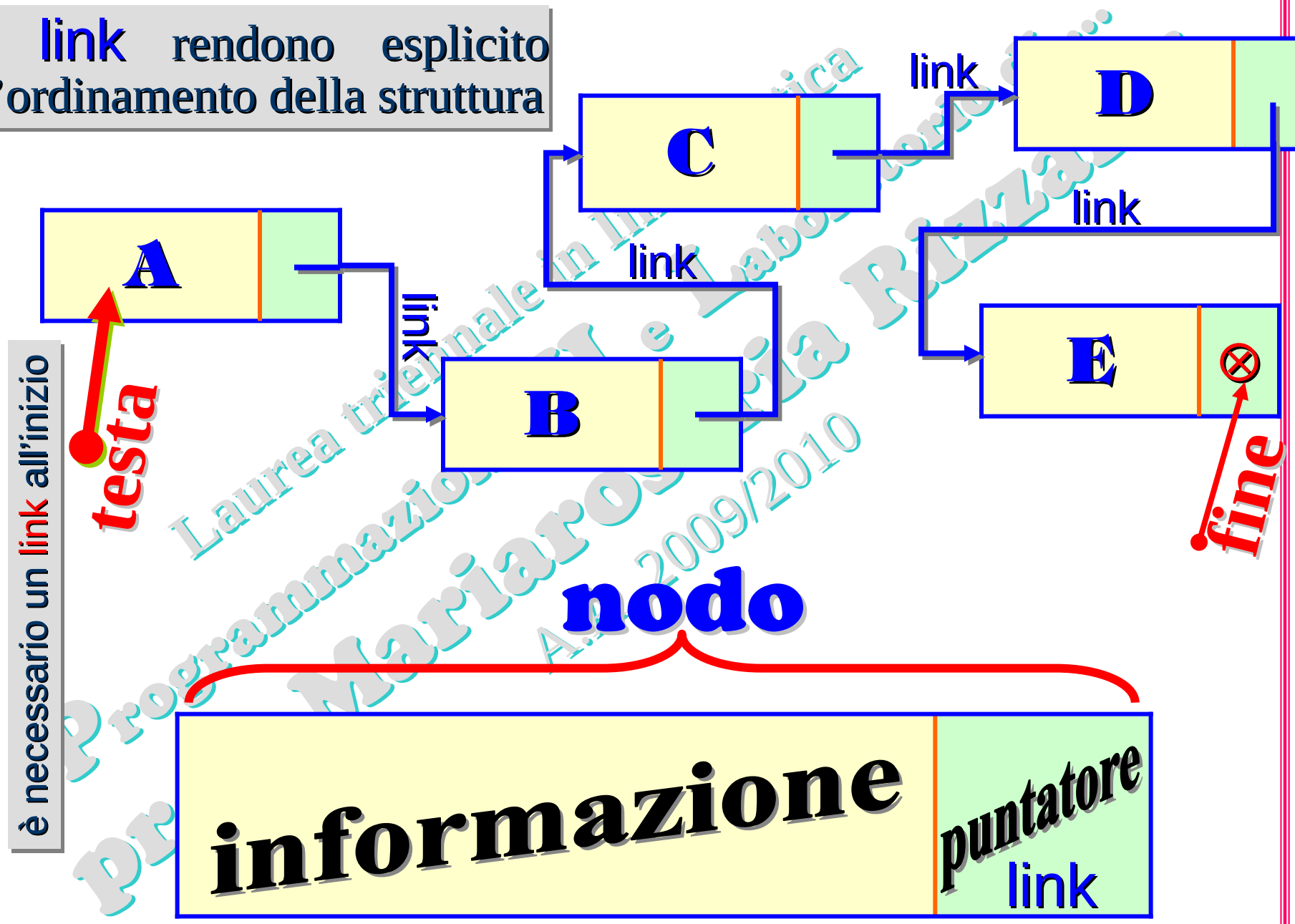
La lista è una struttura lineare **aperta** o **chiusa** in cui l'inserimento e l'eliminazione delle componenti possono avvenire in una qualunque posizione.



La lista è una struttura **ordinata** perché l'accesso ad un elemento può avvenire solo dopo aver visitato tutte gli elementi che lo precedono.

Lista Lineare

I link rendono esplicito l'ordinamento della struttura

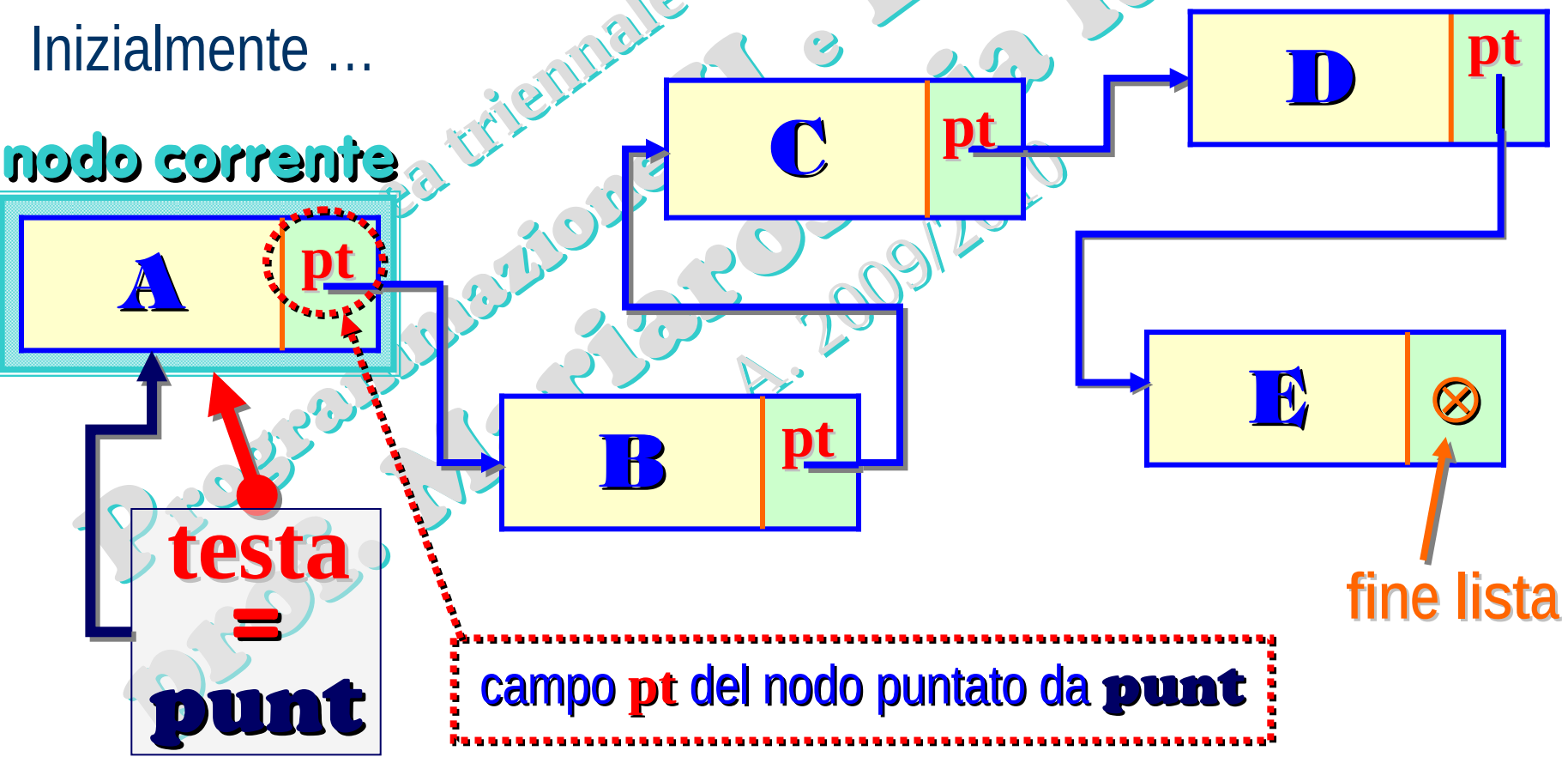


Operazioni sulla lista: visita

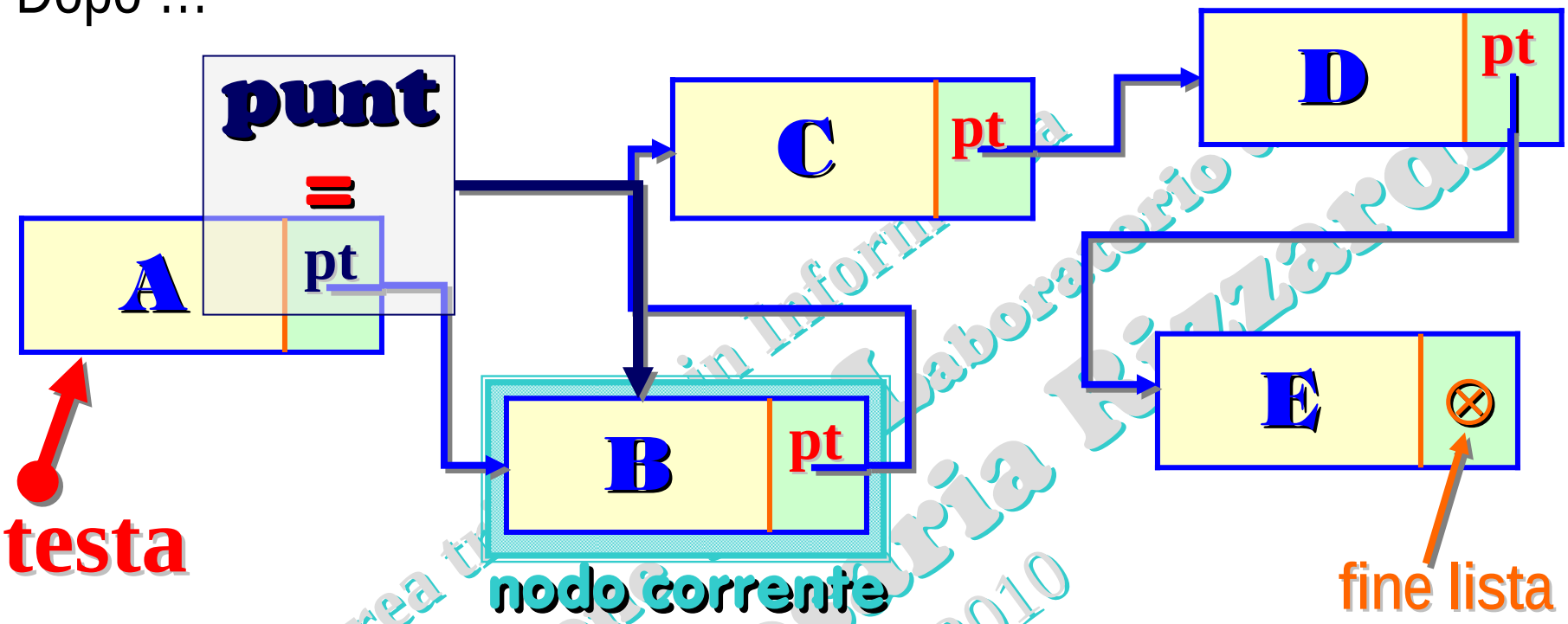
Le variabili che rappresentano la lista sono
i vari nodi ed il puntatore all'inizio della lista (**testa**)

Serve almeno un'altra variabile puntatore (**punt**) [variabile di lavoro]
che di volta in volta punta al nodo corrente

Inizialmente …



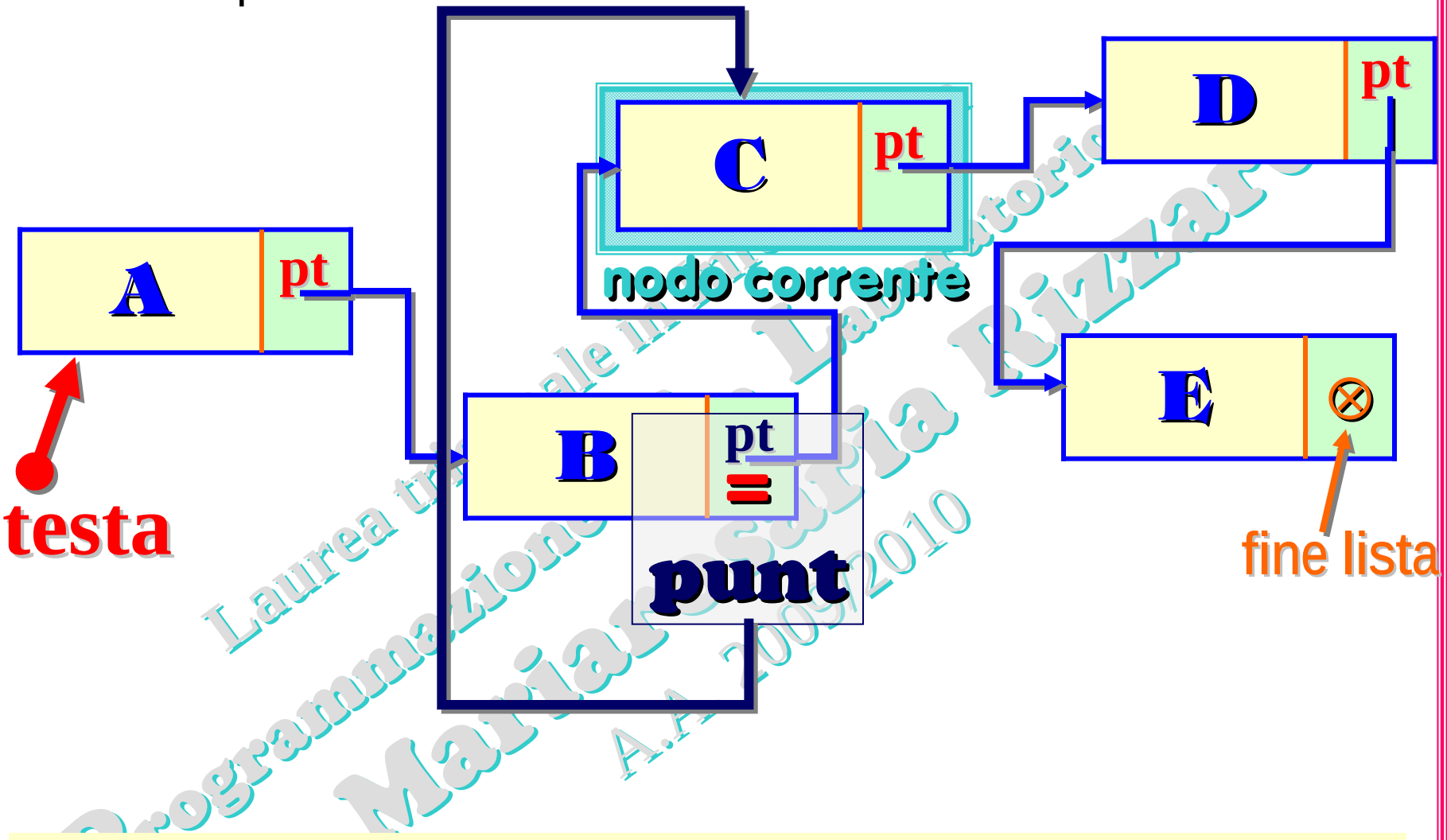
Dopo ...



Il passaggio dal nodo corrente della lista al successivo (avanzamento) avviene solo tramite il valore del suo campo puntatore:

punt ← campo pt del nodo puntato da punt

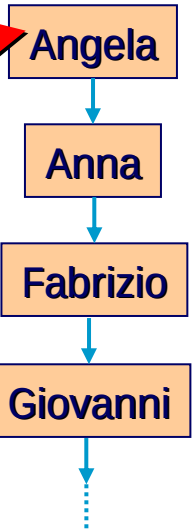
Ancora dopo …



In una lista l'ordinamento logico dei nodi (stabilito tramite il campo puntatore) è ***esplicito*** ed indipendente da quello fisico (legato invece all'allocazione in memoria dei nodi).

Esercizio

Simulare in ***C*** l'algoritmo di visita di una ***linked list*** già memorizzata come array statico di struct (come nella tabella sotto): il primo campo contiene le informazioni (può essere a sua volta una struct), il secondo campo contiene il link (che in questo caso contiene l'indice alla prossima componente).

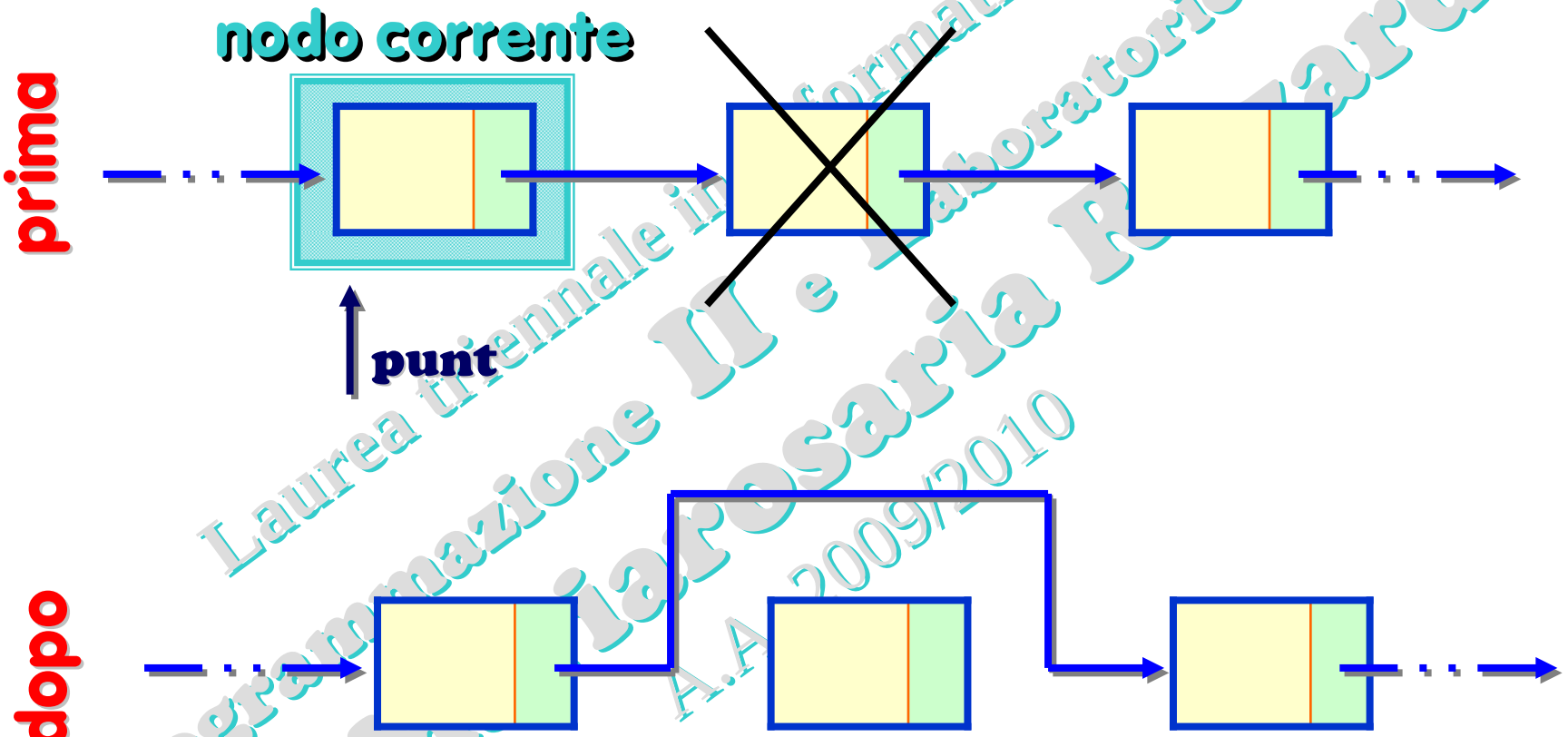


visitare la lista visualizzando le informazioni

equivale a produrre l'elenco

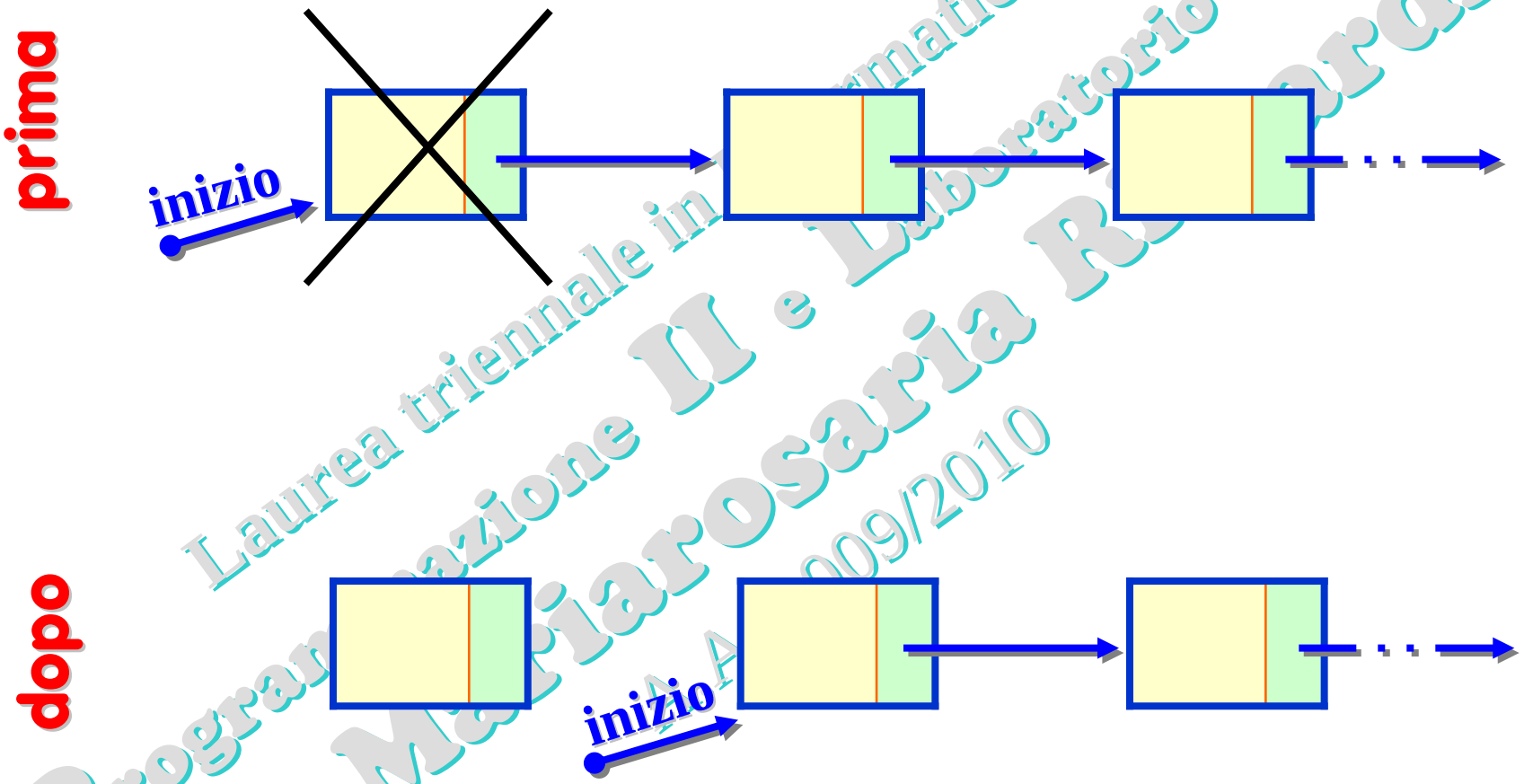
Angela
Anna
Fabrizio
Giovanni
Giuseppe
Marianna
Mario
Patrizia
Sara
Valentina
Valeria

Operazioni sulla lista lineare: eliminazione



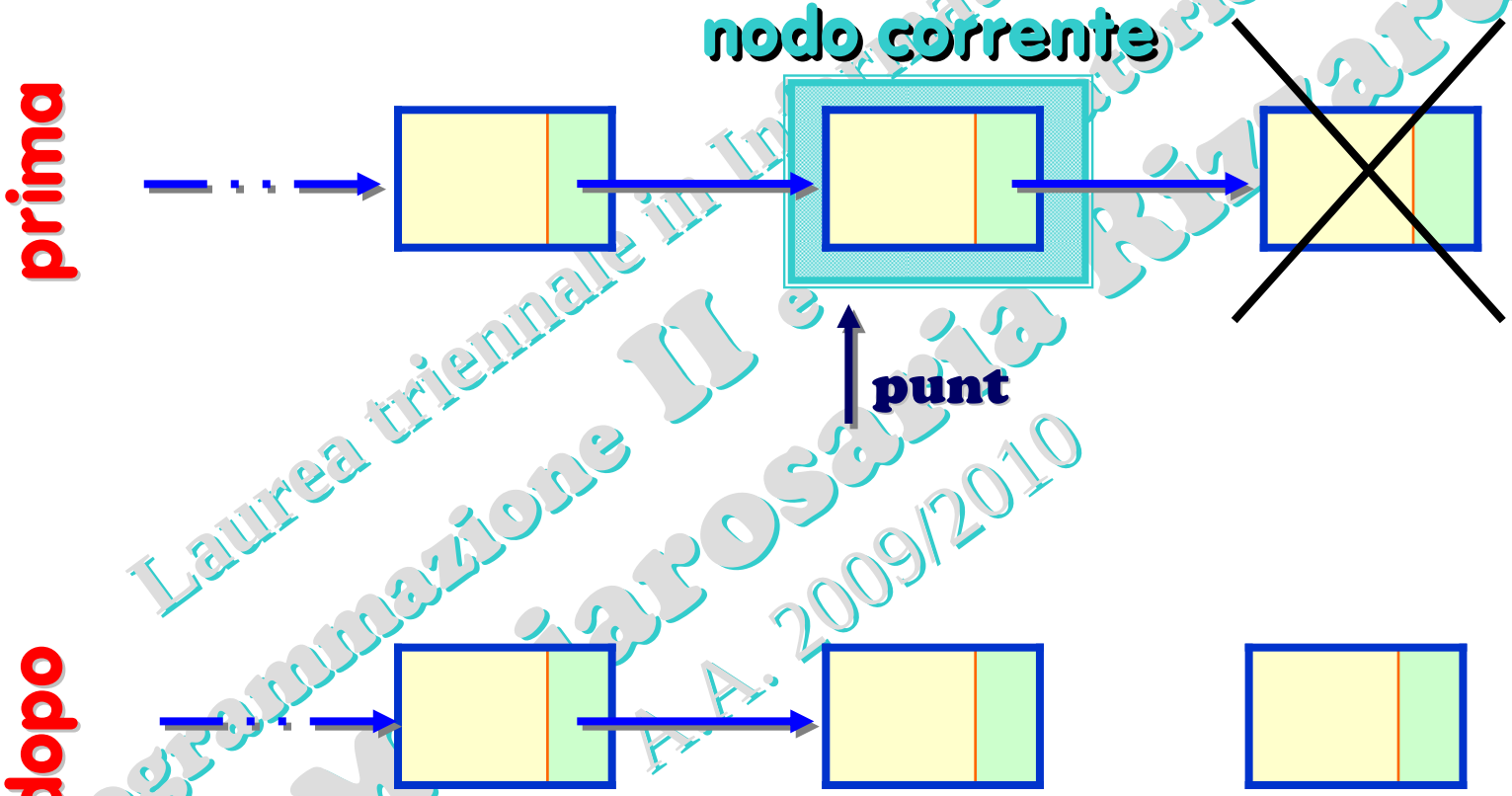
Per eliminare l'elemento corrente, bisogna modificare il link del predecessore affinché punti al successore.

Operazioni sulla lista lineare: eliminazione in testa



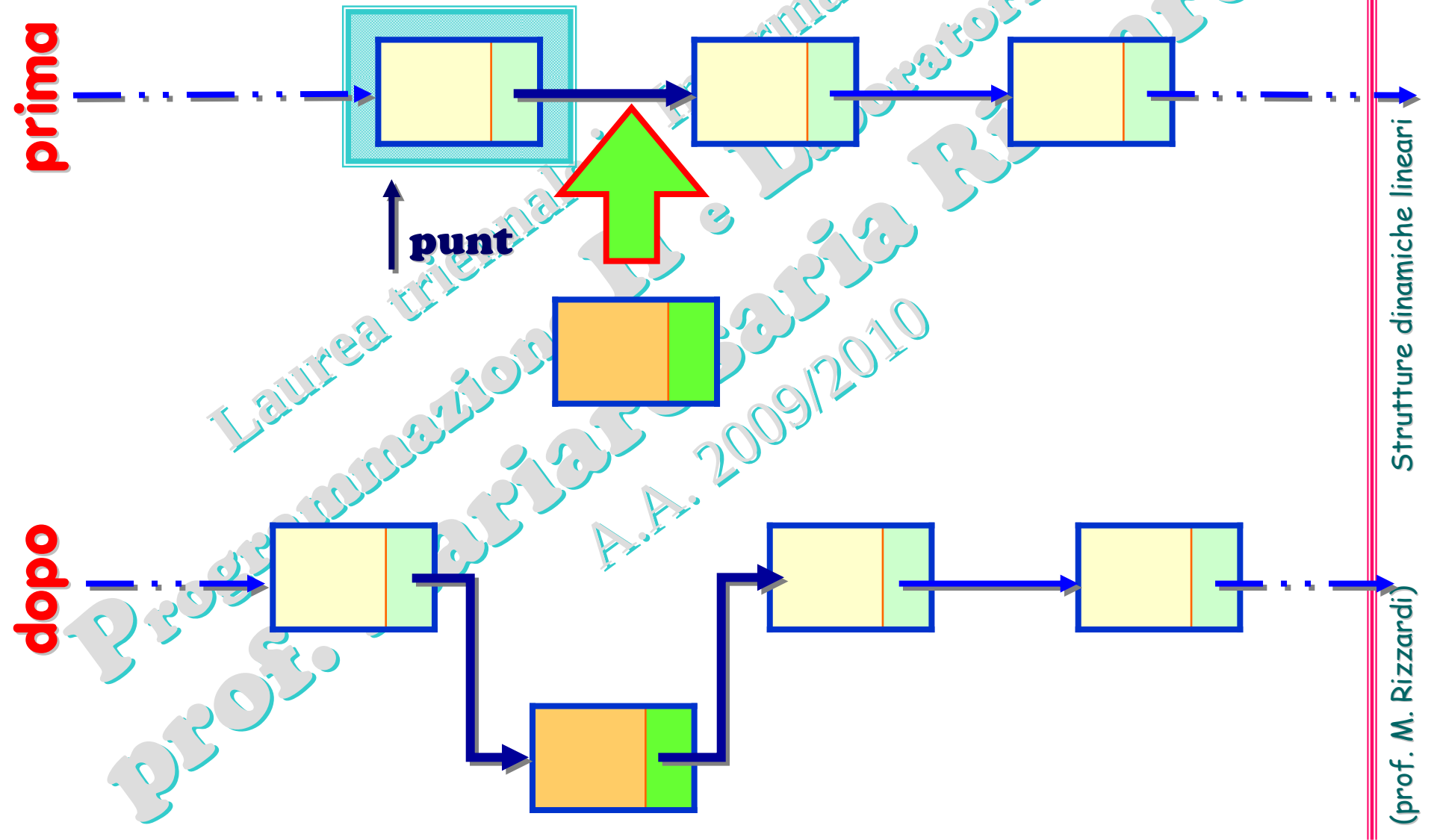
Per eliminare l'elemento in testa, bisogna modificare il link che punta all'inizio della struttura.

Operazioni sulla lista lineare: eliminazione in coda

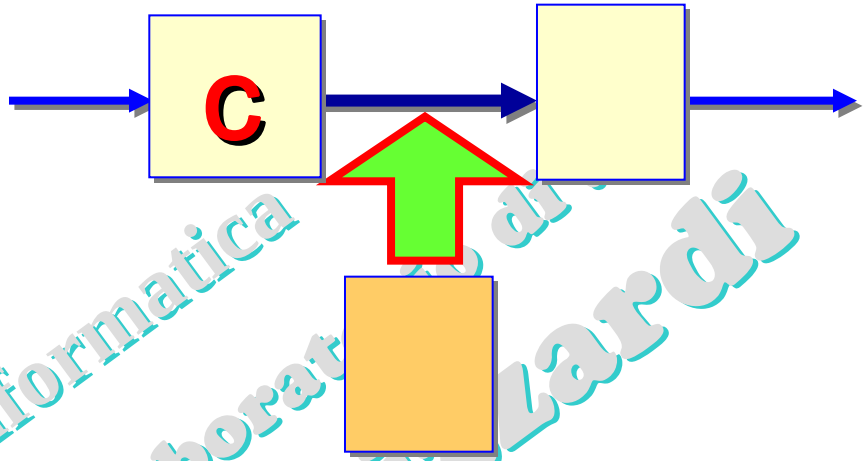


Per eliminare l'elemento in coda, bisogna eliminare il link che punta alla fine della struttura.

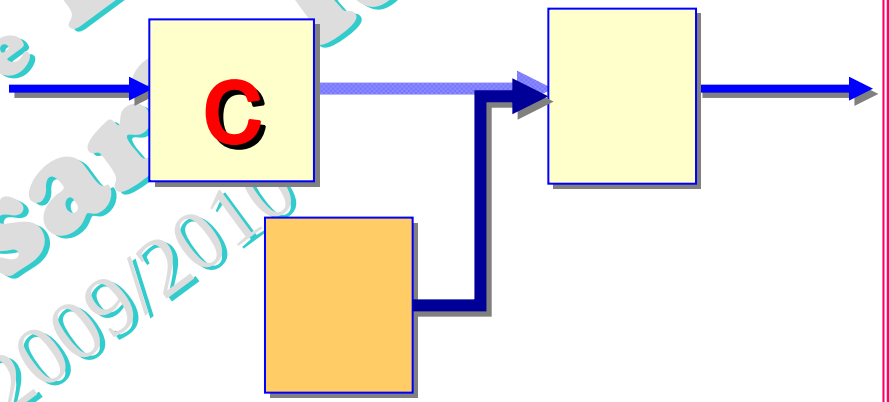
Operazioni sulla lista lineare: inserimento



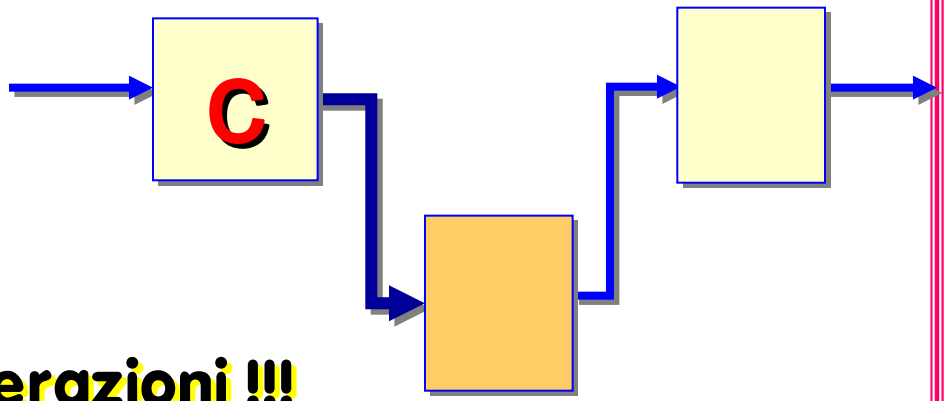
Per inserire un elemento dopo quello corrente, bisogna:



1) inserire il link che va dall'elemento nuovo al successore.

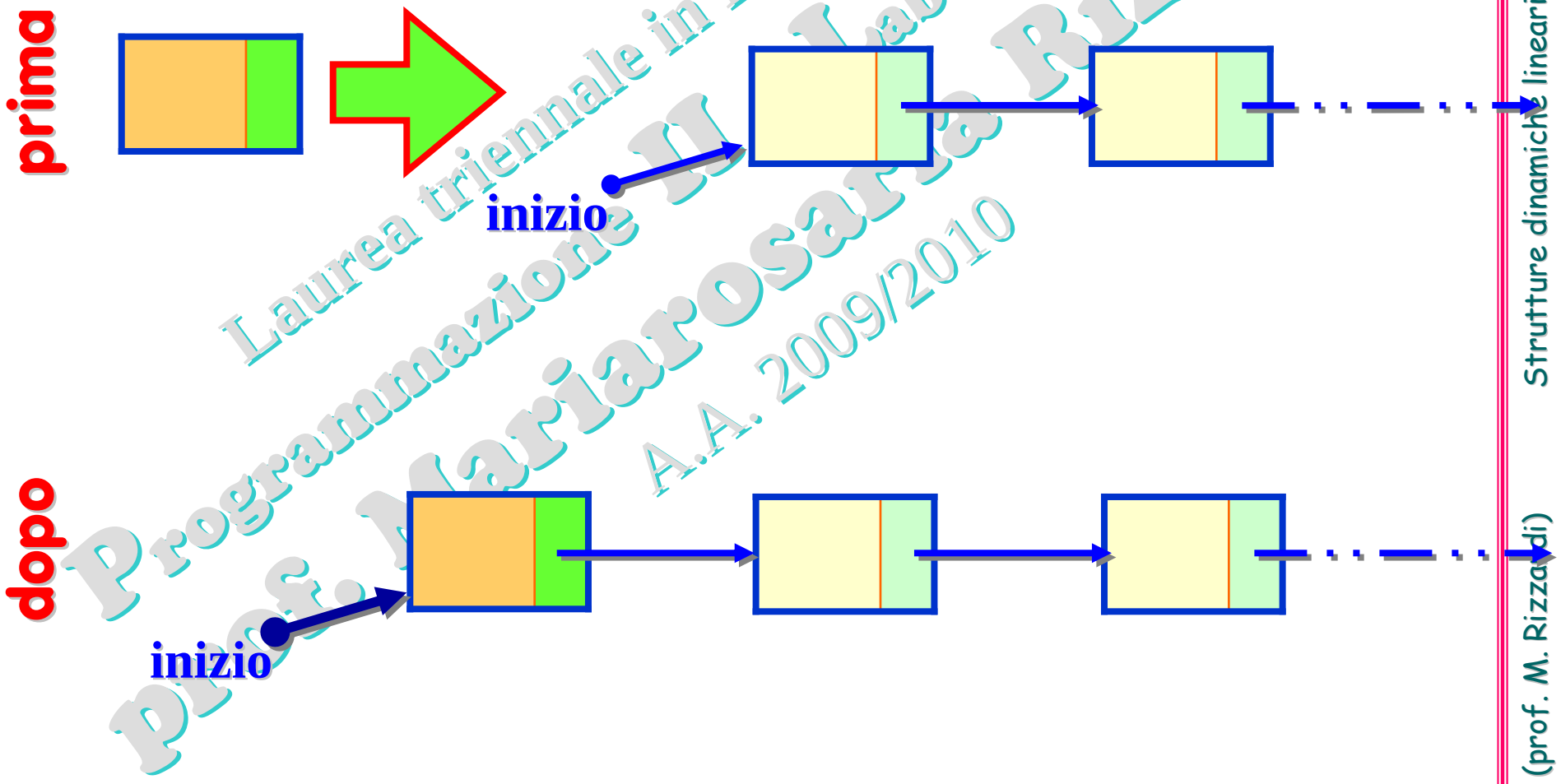


2) modificare il link dell'elemento corrente affinchè punti al nuovo elemento.

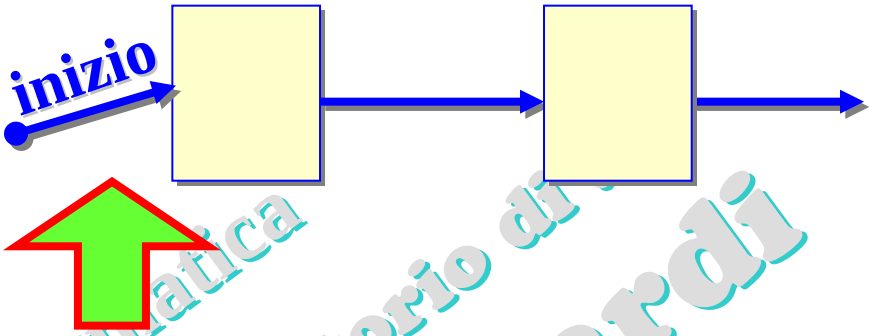


Attenzione all'ordine delle operazioni !!!

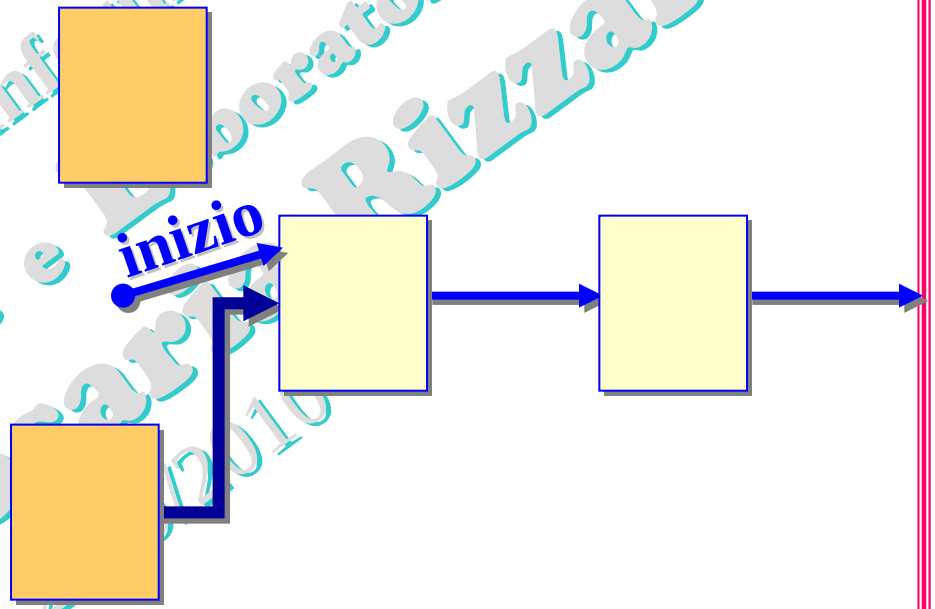
Operazioni sulla lista lineare: inserimento in testa



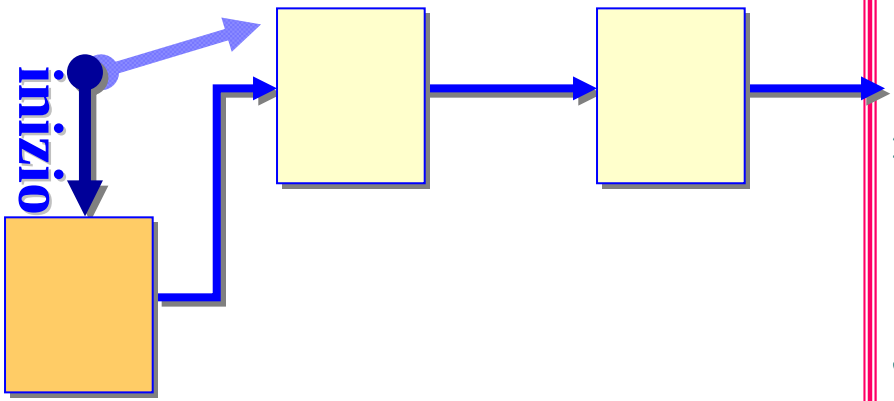
Per inserire un elemento in testa alla struttura, bisogna:



1) inserire il link che va dall'elemento nuovo alla testa della struttura.

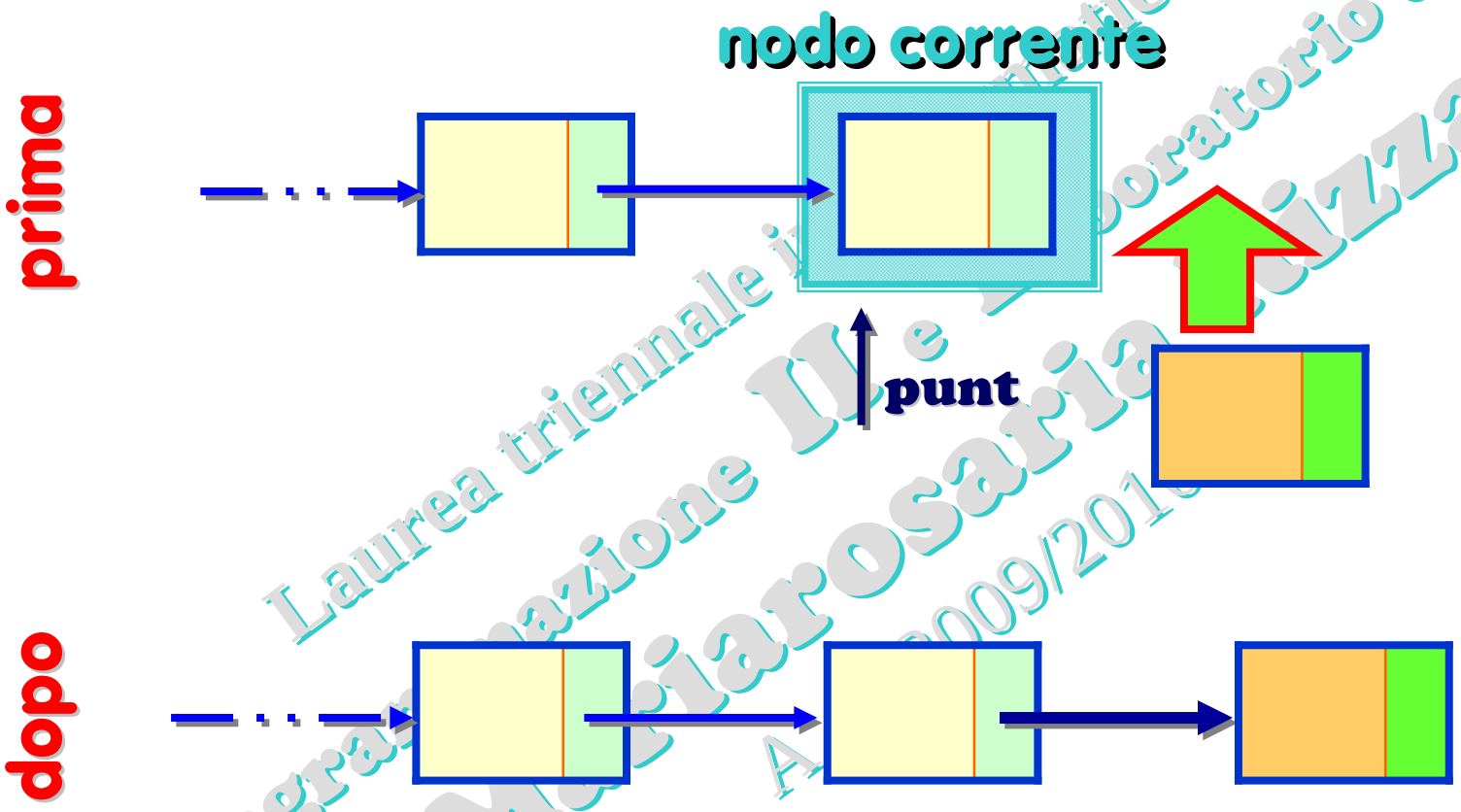


2) modificare il link alla testa affinchè punti al nuovo elemento.



Attenzione all'ordine delle operazioni !!!

Operazioni sulla lista lineare: inserimento in coda



Per inserire un elemento in coda, bisogna aggiungere il link che punta dalla fine della struttura al nuovo elemento.

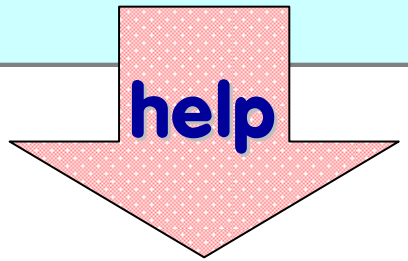
Laboratorio:

Gestione delle camere d'albergo.

Simulare in *C* la *linked list* tramite due array statici: il primo contenente le informazioni (può essere anche un array di struct), il secondo array contenente i link (puntatori ai nodi della lista).

[Suggerimento: L'array di struct corrisponde alla memoria in cui allocare la lista delle camere libere (`ListaLibera`) e la lista delle camere occupate (`ListaDati`). È necessario creare prima la `ListaLibera`, inizializzando l'array dei link in modo che ogni componente punti alla componente successiva. Ogni nodo da inserire nella `ListaDati`, quando una camera viene assegnata ad un cliente, è prelevato dalla testa della `ListaLibera` ed inserito nella testa della `ListaDati`; mentre il nodo da eliminare dalla `ListaDati`, quando una particolare camera viene liberata, è restituito alla `ListaLibera` (in testa) per poter essere riutilizzato in seguito.]

[liv. 3]



L'array di struct NODO {INFO, pnext} simula la memoria dove saranno allocati i nodi della lista lineare ListaDati.

Per rendere dinamica la gestione della "memoria simulata" viene creata inizialmente la lista lineare ListaLibera (con p_Libera puntatore alla sua testa)

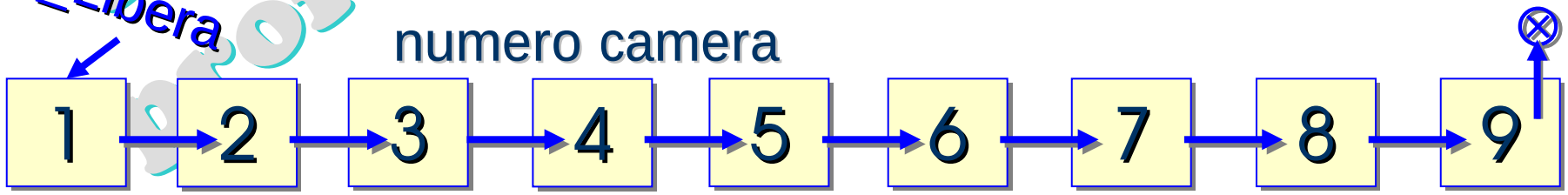
ListaLibera inizializzata come in tabella

[cioè ogni nodo punta alla componente che segue]

NODO

	INFO	pnext
0	1	1
1	2	2
2	3	3
3	4	4
4	5	5
5	6	6
6	7	7
7	8	8
8	9	−1=fine

p_Libera



sono necessari due puntatori alla testa delle liste:

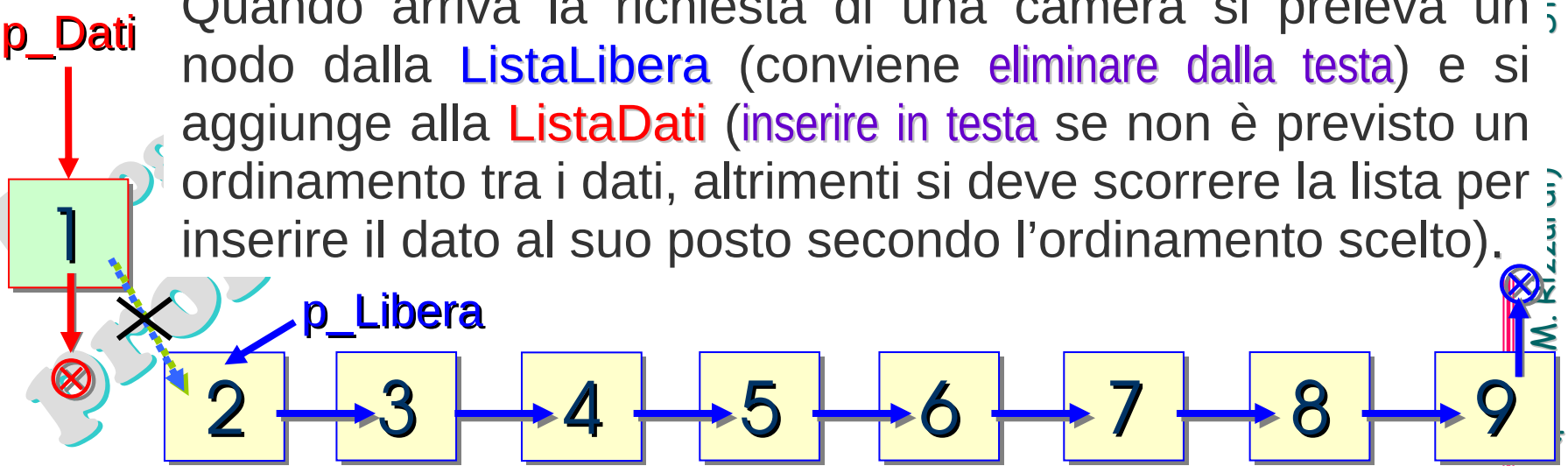
p_Libera: punta alla testa della ListaLibera

p_Dati: punta alla testa della ListaDati

Le due liste ListaLibera e ListaDati condividono la stessa memoria (array NODO). Inizialmente ListaDati è vuota.

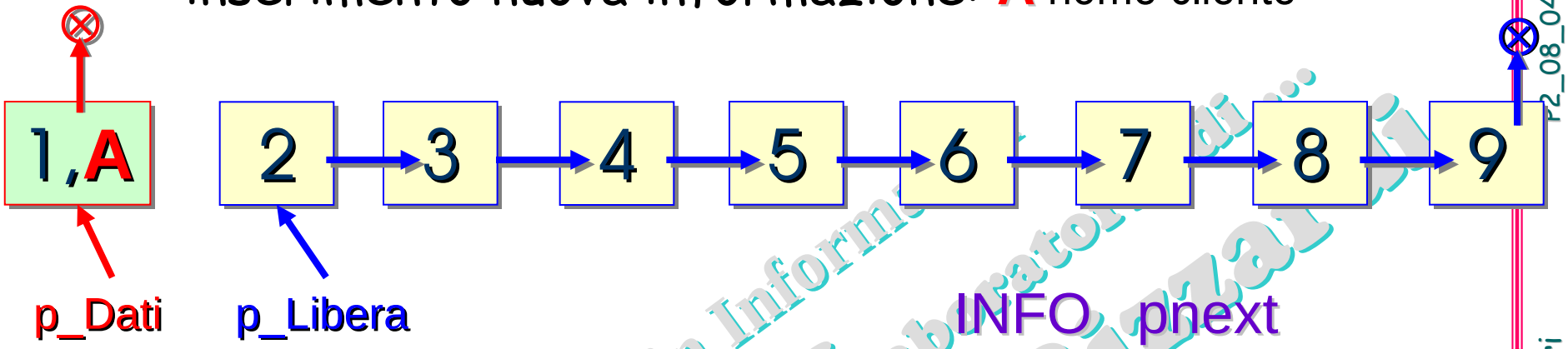
Situazione iniziale

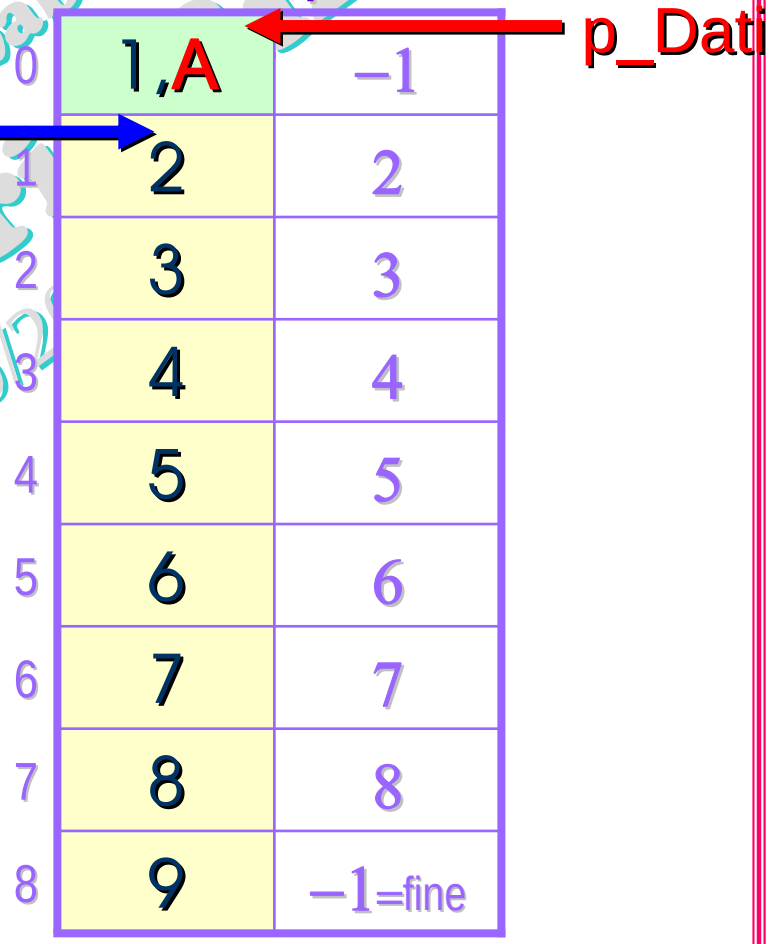
Richiesta di una camera



Quando arriva la richiesta di una camera si preleva un nodo dalla ListaLibera (conviene eliminare dalla testa) e si aggiunge alla ListaDati (inserire in testa se non è previsto un ordinamento tra i dati, altrimenti si deve scorrere la lista per inserire il dato al suo posto secondo l'ordinamento scelto).

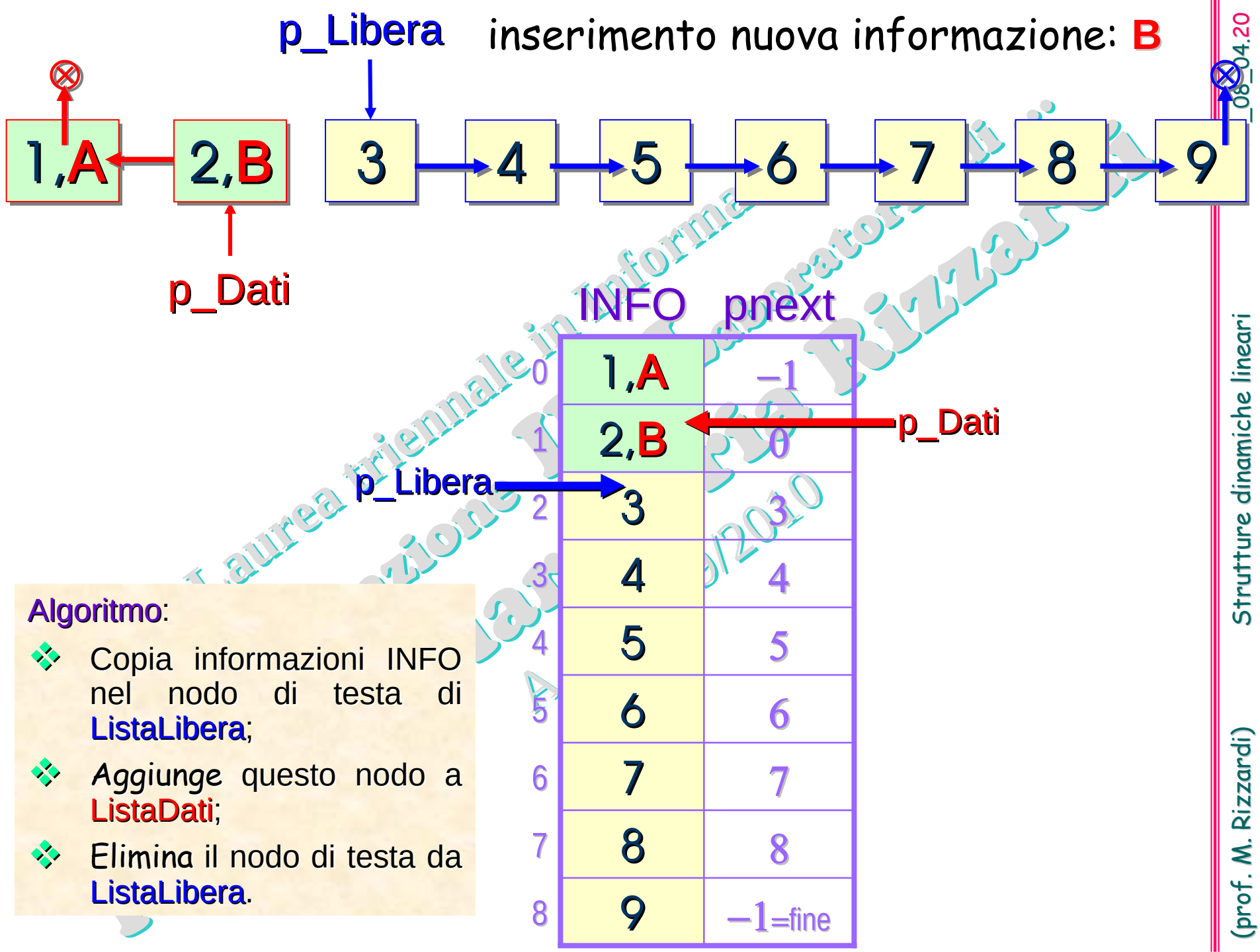
inserimento nuova informazione: A nome cliente





Algoritmo:

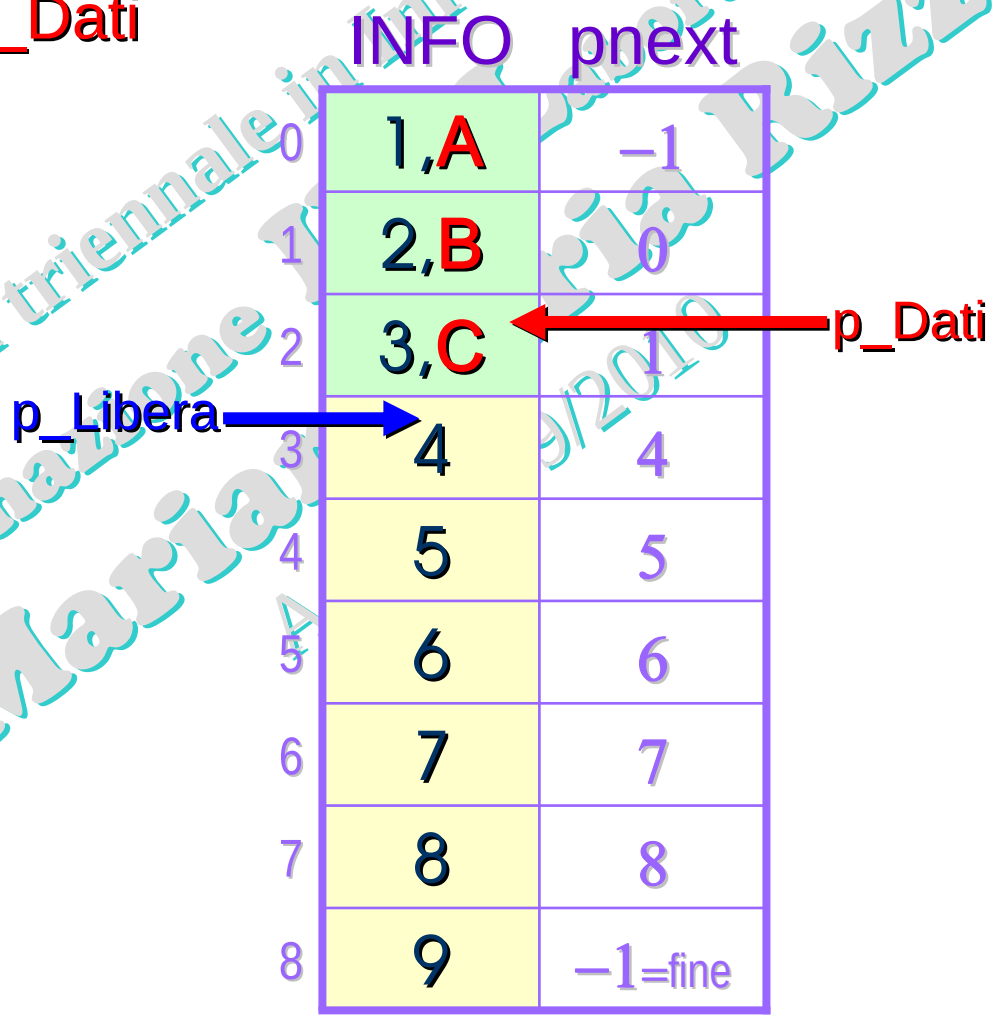
- Copia informazioni INFO nel nodo di testa di ListaLibera;
- *Aggiunge* questo nodo a ListaDati;
- Elimina il nodo di testa da ListaLibera.



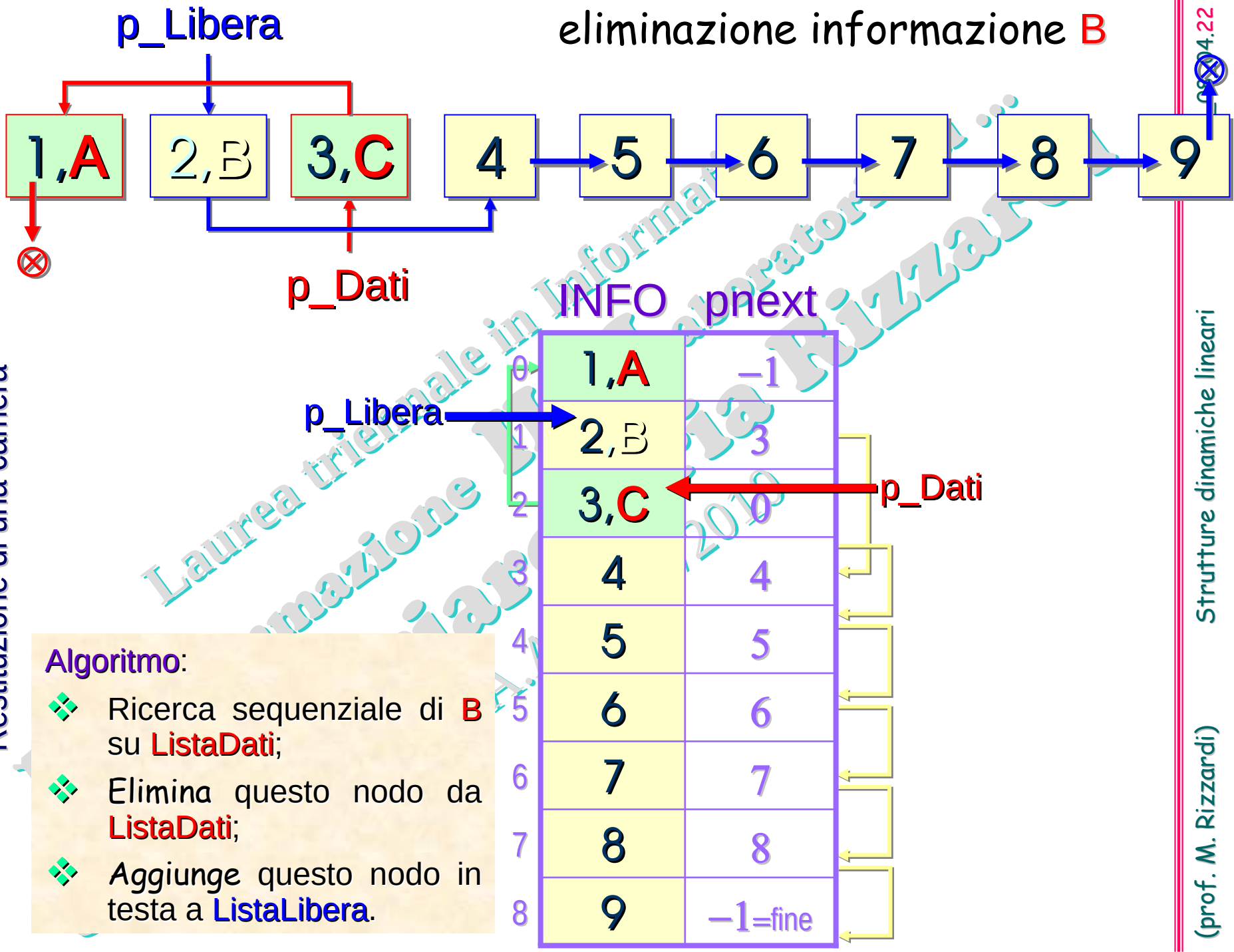
Algoritmo:

- Copia informazioni INFO nel nodo di testa di ListaLibera;
- *Aggiunge* questo nodo a ListaDati;
- Elimina il nodo di testa da ListaLibera.

inserimento nuova informazione



eliminazione informazione B



Algoritmo:

- Ricerca sequenziale di **B** su **ListaDati**;
- *Elimina* questo nodo da **ListaDati**;
- *Aggiunge* questo nodo in testa a **ListaLibera**.