CORSO DI PROGRAMMAZIONE CLA
A.A. 2019-20

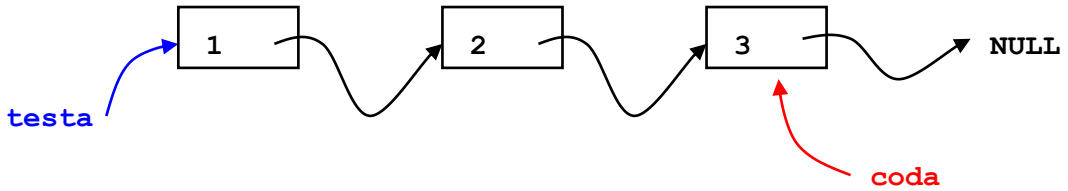
Dispensa 22

Dott. Filippo Piccinini
e-mail: f.piccinini@unibo.it

Ringrazio sinceramente il Prof. Mirko Ravaioli per avere gentilmente condiviso il materiale usato negli anni precedenti.

22 Le Code

Una coda è una lista con un puntatore alla testa dell'elenco delle celle e un puntatore all'ultimo elemento dell'elenco:



La struttura dati per gestire ogni elemento (cella) della coda, come per le liste è:

```
struct cella{
      int valore;
      struct cella *next;
};
```

Ogni cella, come per le liste, ha un puntatore all'elemento successivo.
Per gestire una coda dovremo avere due variabili (puntatori): il puntatore al primo elemento (testa) e il puntatore all'ultimo elemento (coda):

```
struct cella *testa = NULL; //puntatore al primo elemento
struct cella *coda = NULL; //puntatore all'ultimo elemento
```

In una coda:

- tutti i nuovi elementi vengono inseriti in coda, quindi in fondo all'elenco,
- tutte le operazioni che implicano una lettura (stampa, cancellazione, etc.) vengono fatte sempre a partire dalla testa,
- ogni volta che si va a considerare un elemento questo viene tolto.

La coda è una struttura dati di tipo FIFO (First In First Out) dove il primo elemento inserito è anche il primo ad essere "gestito". Si immagini una fila di persone davanti ad uno sportello postale: il primo che arriva è anche il primo ad essere servito! Vedere il codice nella cartella "coda_funzioni_ver1" per un esempio di coda realizzato con la struttura sopra descritta e i due puntatori "*testa" e "*coda".

Dato che per gestire la coda sono sempre necessari i due puntatori descritti sopra, solitamente si preferisce raggruppare il tutto all'interno di un ulteriore struttura:

```
struct coda{
      struct cella *primo; //o anche testa
      struct cella *ultimo; //o anche coda
};
typedef struct coda coda;

…nel main()…
coda Coda1;
```

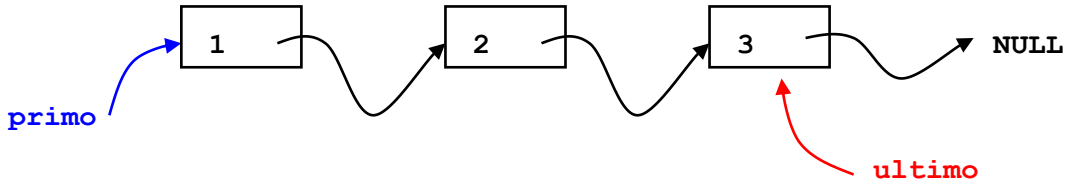
"Coda1" è una variabile di tipo "struct coda", quindi accedendo a "Coda1->primo" si ottiene il puntatore al primo elemento della coda, mentre con "Coda1->ultimo" si accede all'ultimo elemento della coda. Vedere il codice nella cartella "coda_funzioni_ver2" per un esempio di coda realizzato con la "struct coda" sopra descritta.

22.1 Inserimento di un elemento (funzione *push*)

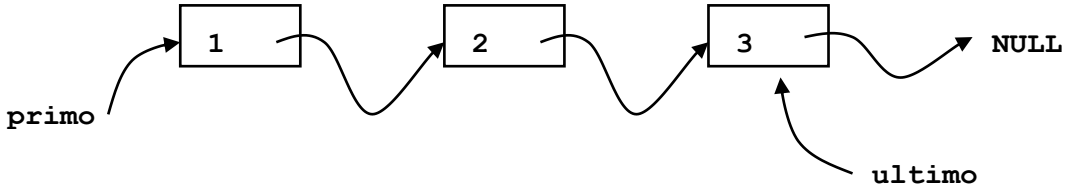
Considerando le seguenti strutture dati e variabili:

```
struct cella{
      int valore;
      struct cella *next;
};

struct coda{
      struct cella *primo; //o anche testa
      struct cella *ultimo; //o anche coda
};
typedef struct coda coda;

…nel main()…
coda Coda1;
(&Coda1)->primo=NULL;
(&Coda1)->ultimo=NULL;
```

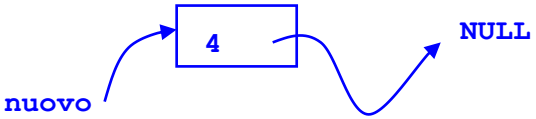
Dove "cella" è la struttura che definisce come è fatto ogni elemento della coda, "Coda1" è la variabile che contiene il puntatore al primo e all'ultimo elemento della lista



Attraverso una chiamata alla funzione *malloc()* allochiamo in memoria lo spazio necessario per mantenere la nuova cella da inserire all'interno della lista:



1

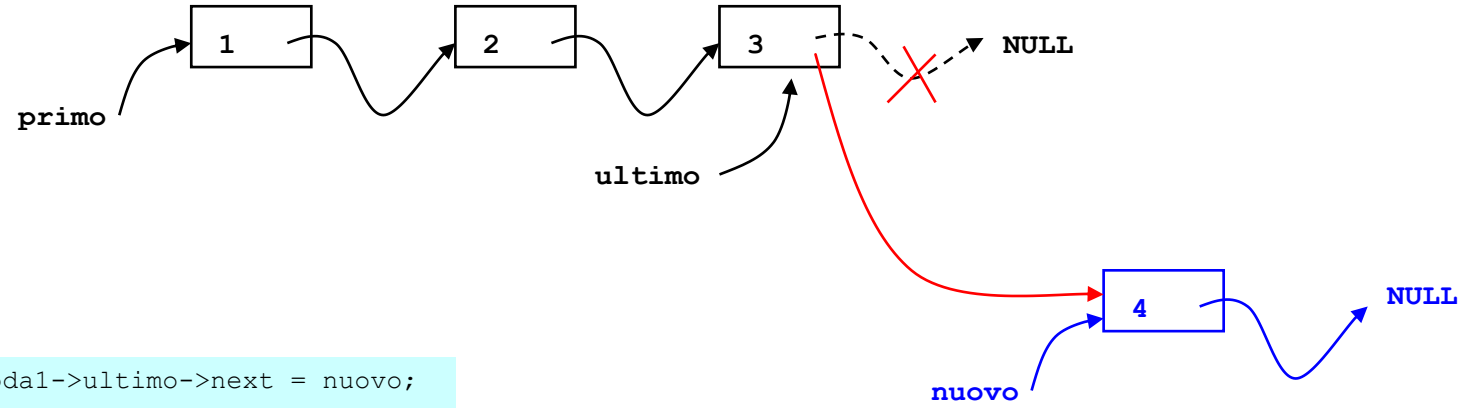



```
nuovo = (struct cella*)malloc(sizeof(struct cella));
```

"nuovo" è un puntatore ad un elemento di tipo "`struct cella`" allo stesso modo di "primo", la funzione *malloc()* restituisce l'indirizzo di memoria della prima cella allocata. Tale indirizzo viene memorizzato all'interno della variabile (puntatore) "nuovo".

La prima operazione da compiere è collegare l'ultima cella della coda alla nuova, quindi fare in modo che l'elemento "next" dell'ultimo elemento punti al nuovo elemento creato:

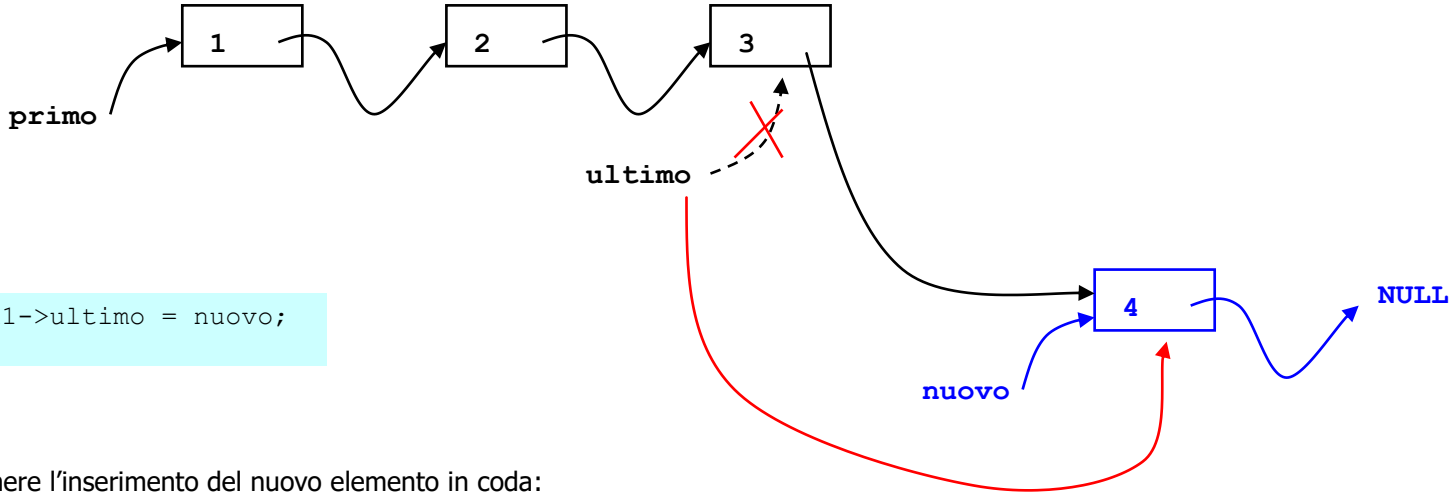
2




```
Coda1->ultimo->next = nuovo;
```

La seconda operazione consiste nell'associare il puntatore "ultimo" al nuovo elemento creato:

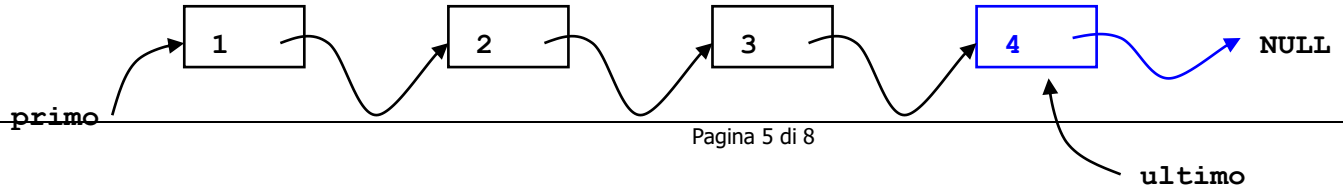
3




```
Coda1->ultimo = nuovo;
```

In modo da ottenere l'inserimento del nuovo elemento in coda:

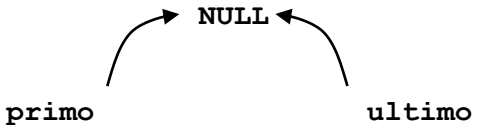
4



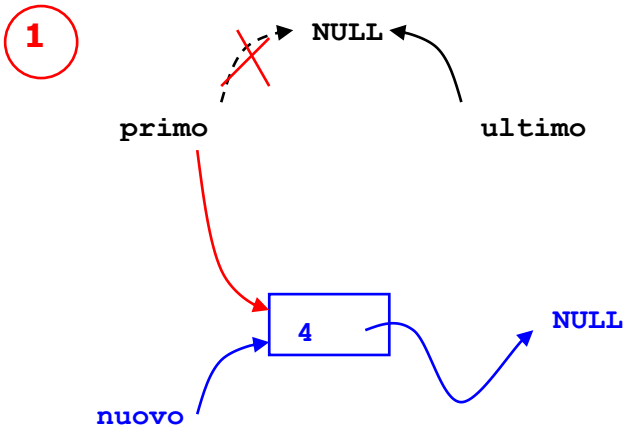
Riassumendo il codice necessario per l'inserimento di un elemento in coda:

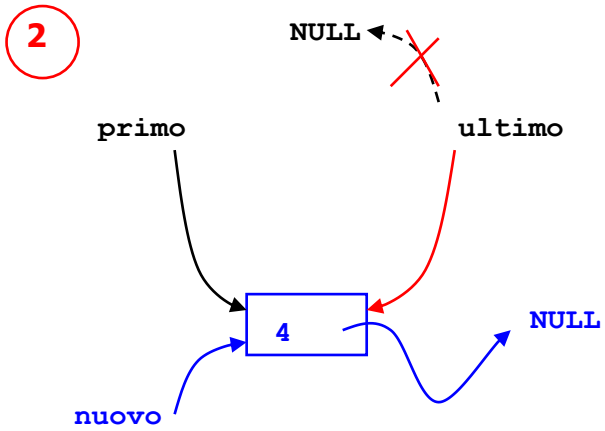
```
nuovo = (struct cella*)malloc(sizeof(struct cella));
Coda1->ultimo->next = nuovo;
Coda1->ultimo = nuovo;
```

ATTENZIONE! Le istruzioni descritte NON funzionano se la coda è vuota! Se la coda è vuota non basterà aggiustare il puntatore all'ultimo elemento, ma dovremo sistemare anche il puntatore al primo elemento in coda. Questo perchè essendo la coda vuota, il nuovo elemento inserito risulterà sia il primo che l'ultimo!
Considerando quindi una coda vuota:



Avremo sia il primo che l'ultimo puntatore settati al valore NULL. Il nuovo elemento da inserire sarà puntato sia dalla testa che dalla coda:



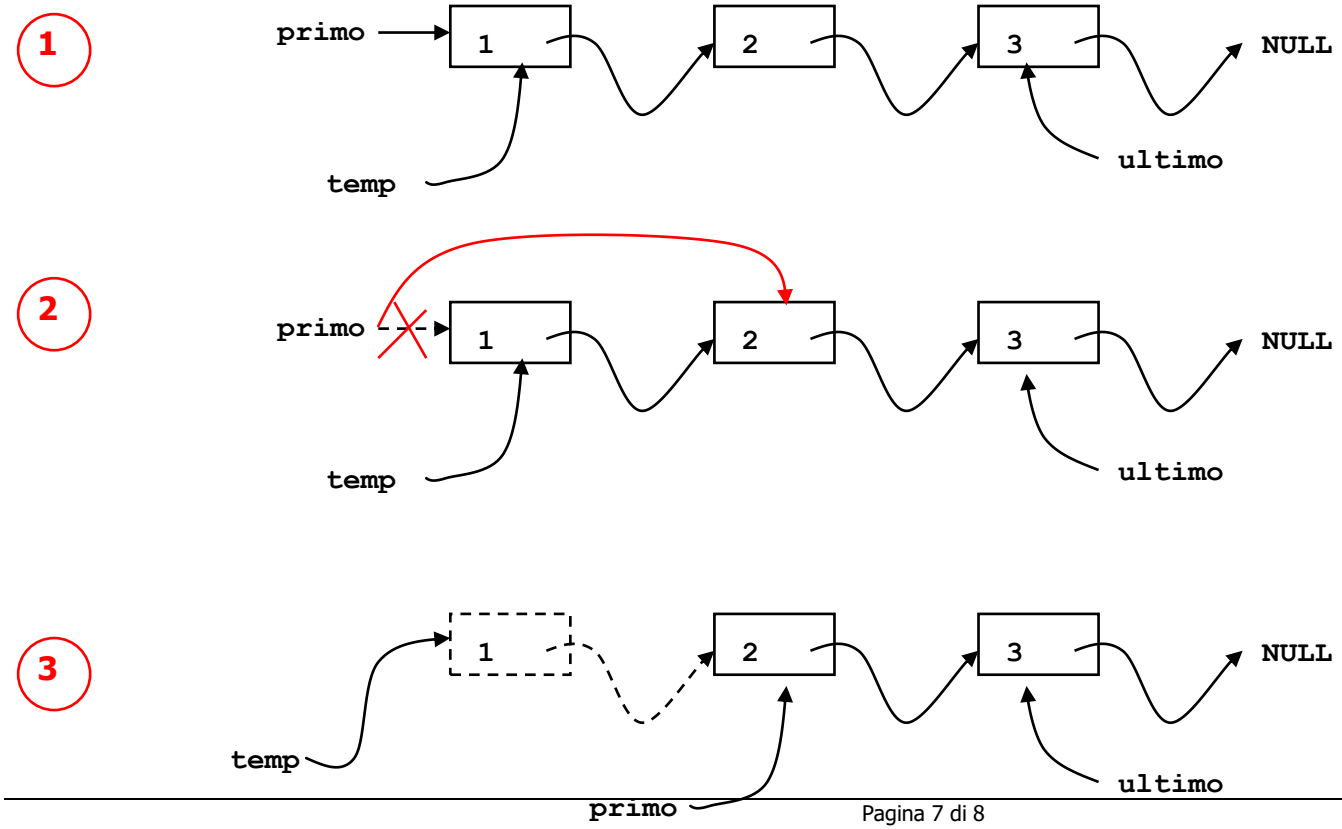


Riassumendo il codice necessario per l'inserimento di un elemento in coda, sempre valido sarà:

```
nuovo = (struct cella*)malloc(sizeof(struct cella));
if (Coda1->primo == NULL) //se la coda è vuota
      Coda1->primo = nuovo;
else
      Coda1->ultimo->next = nuovo;
Coda1->ultimo = nuovo;
```

22.2 Lettura di un elemento (funzione *pop*)

Per leggere l'elemento in testa, il riferimento alla testa della lista deve passare al secondo elemento per evitare di perdere l'intera coda:

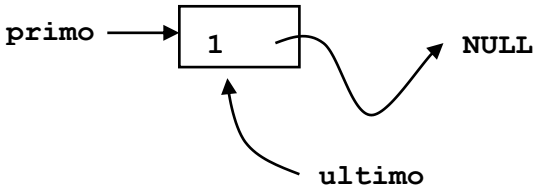


Viene utilizzato un puntatore "temp" per tener traccia del primo elemento prima di far puntare a "testa" all'elemento successivo:

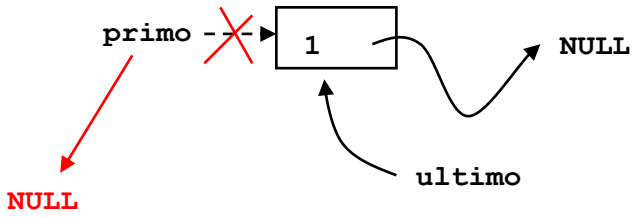
```
temp = Coda1->primo;
Coda1->primo = Coda1->primo->next;
temp->next = NULL;
```

ATTENZIONE! Se l'elemento è l'unico in coda bisogna reimpostare anche il puntatore all'ultimo elemento:

1

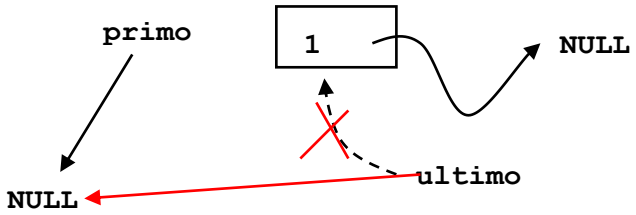


2



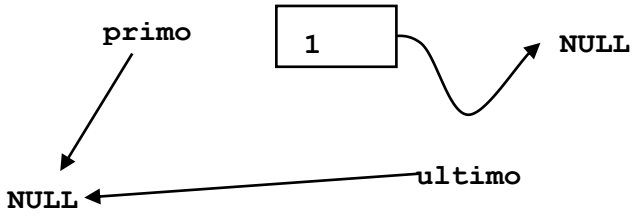

```
Coda1->primo = NULL;
```

3




```
Coda1->ultimo = NULL;
```

4




```
/**una volta che l'elemento è
stato staccato, viene
restituito dalla funzione pop.
In alternativa può essere
fatta la free dell'elemento**/

free(temp);
```