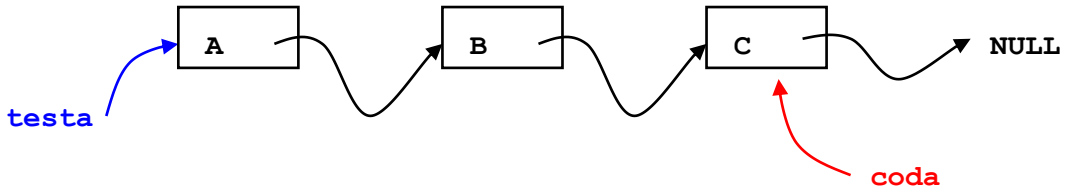
CORSO DI PROGRAMMAZIONE
A.A. 2014-15

Dispensa 23

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

23 Le Code

Una coda in sintesi è una lista (come descritta nella dispensa 17) quindi con un puntatore alla testa dell'elenco delle celle, con l'aggiunta di un puntatore alla all'ultimo elemento dell'elenco:



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

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

Ogni cella come per le liste ha un puntatore all'elemento successivo, ovviamente dovremo avere due variabili (puntatori) aggiuntivi per gestire 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, mentre tutte le altre operazioni di lettura (stampa, cancellazione...) vengono eseguite a partire dalla testa. 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!

Dato che per gestire la coda sono sempre necessari i 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
};

struct Coda coda;
```

dove coda è una variabile di tipo "struct coda", quindi accedendo a "coda.primo" si ottiene il puntatore al primo elemento della coda, mentre con "coda.ultimo" si accede all'ultimo elemento della coda.

23.1 Inserimento di un elemento (push)

Considerando le seguenti strutture dati e variabili:

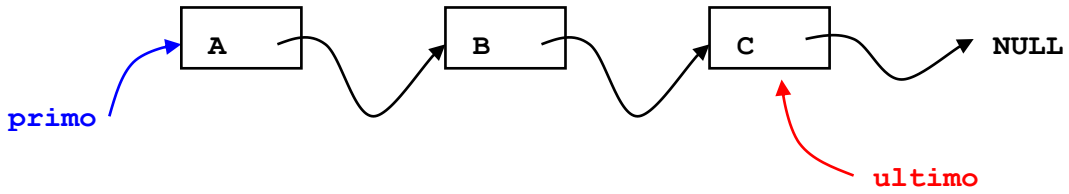
```
struct cella{
      char valore;
      struct cella *next;
};

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

struct Coda coda;
```

coda.primo = NULL;
coda.ultimo = NULL;

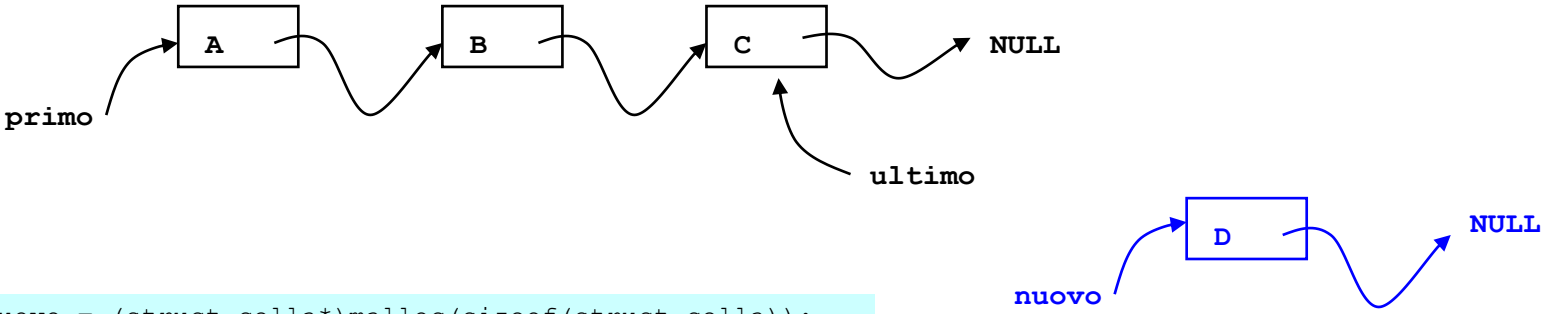
Dove "cella" è la struttura che definisce come è fatto ogni elemento della coda, "coda" è la variabile che contiene il puntatore al primo e all'ultimo elemento della lista (testa mantiene l'indirizzo di memoria della prima cella in lista).



"coda" è una variabile di tipo "struct coda", quindi accedendo a "coda.primo" si ottiene il puntatore al primo elemento della coda, mentre con "coda.ultimo" si accede all'ultimo elemento della coda.

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

1

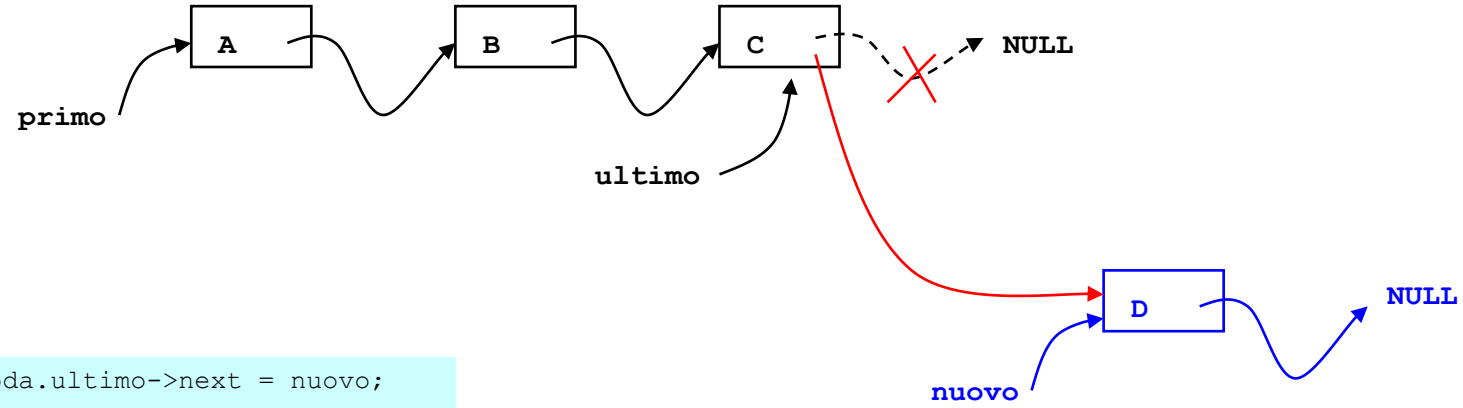



```
nuovo = (struct cella*)malloc(sizeof(struct cella));
```

"nuovo" è un puntatore ad un elemento di tipo "struct cella" allo stesso modo di "testa", 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 al nuovo creato come indicato nel punto sopra:

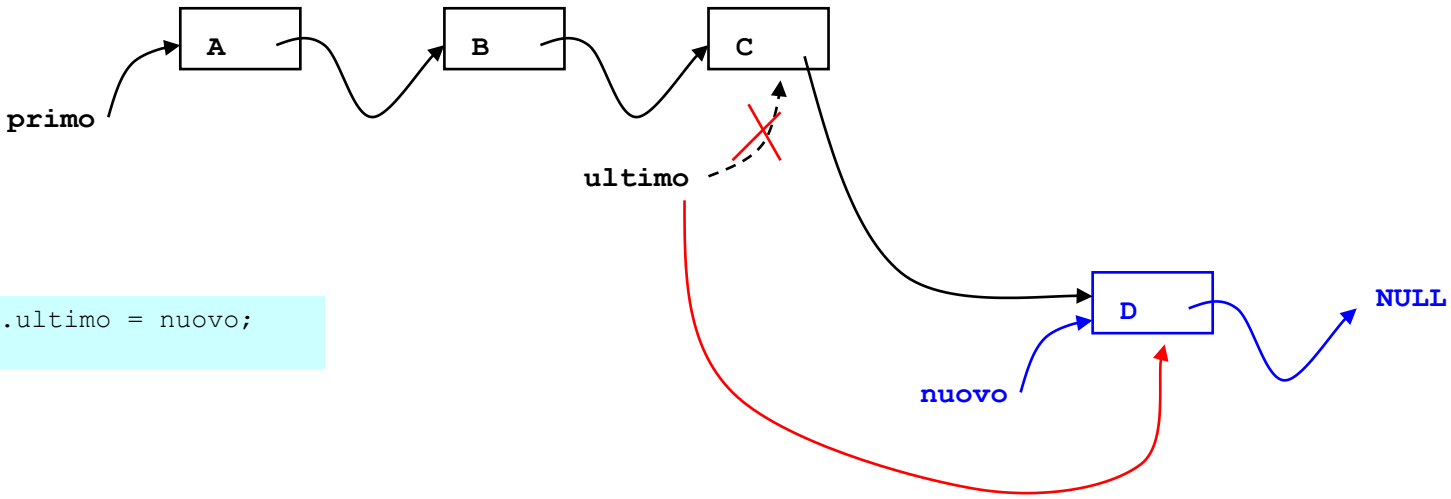
2




```
coda.ultimo->next = nuovo;
```

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

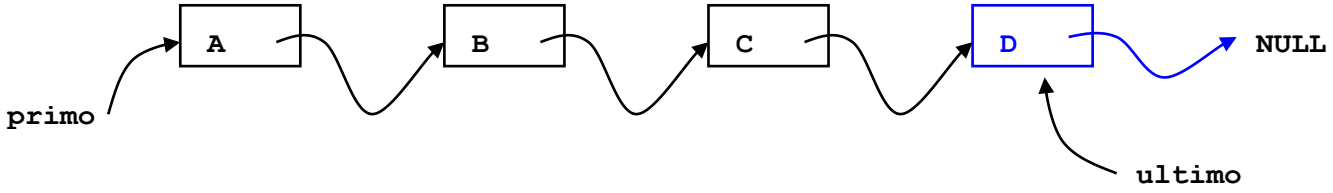
3




```
coda.ultimo = nuovo;
```

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

4

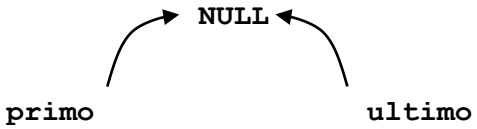


Riassumendo il codice necessario per l'inserimento di un elemento in coda:

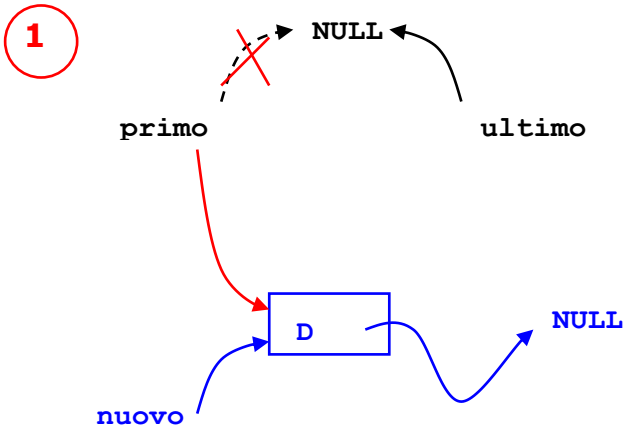
```
nuovo = (struct cella*)malloc(sizeof(struct cella));
coda.ultimo->next = nuovo;
coda.ultimo = nuovo;
```

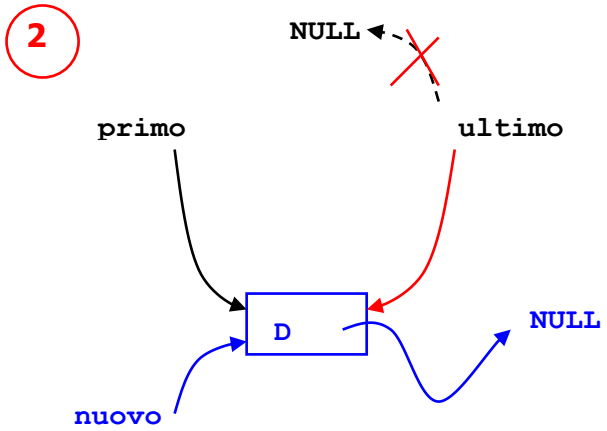
ATTENZIONE! Le istruzioni descritte NON funzionano se la coda è vuota! Se la coda è vuota non basterà aggiustare il puntatore all'ultimo elemento, ma dovremo sistemare anche il puntatore al primo elemento in coda. Questo perchè essendo la coda vuota, il nuovo elemento inserito risulterà sia il primo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

Considerando 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:



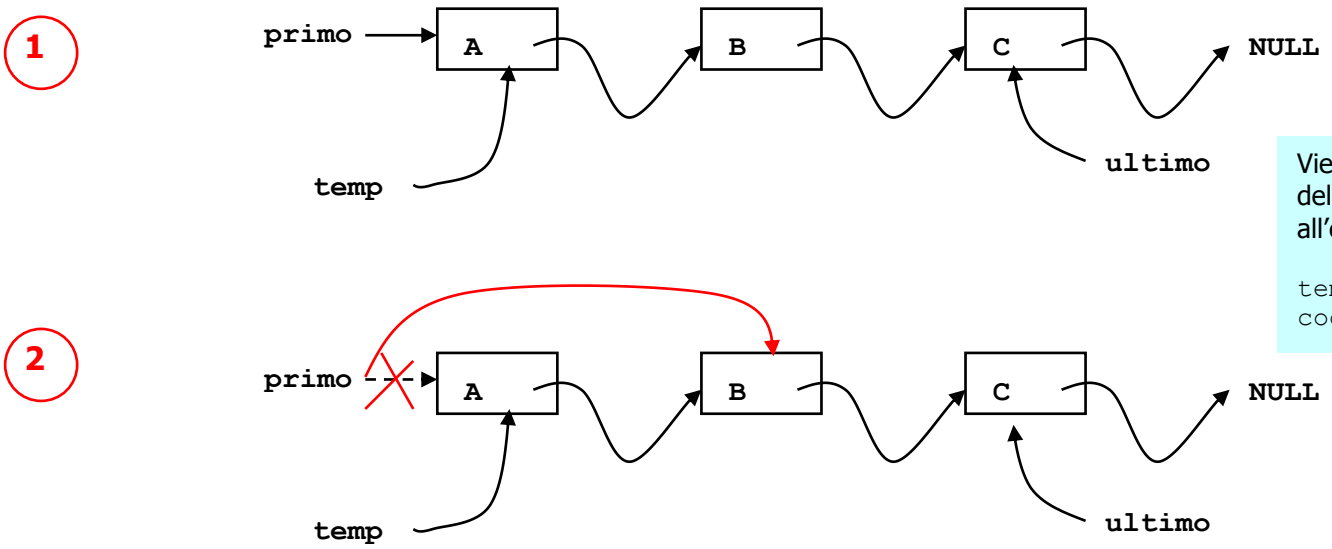


Riassumendo il codice necessario per l'inserimento di un elemento in coda, sempre valido sarà:

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

23.2 Lettura di un elemento (pop)

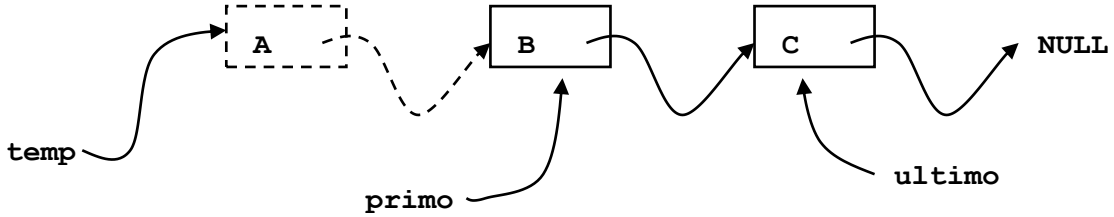
Per leggere l'elemento in testa, il riferimento alla testa della lista deve passare al secondo elemento per evitare di perdere l'intera coda:



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

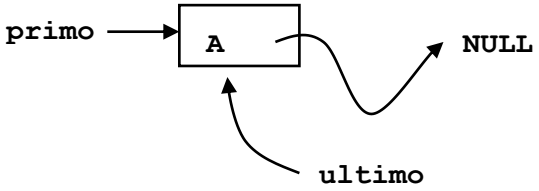
```
temp = coda.primo;
coda.primo = coda.primo->next;
```

3

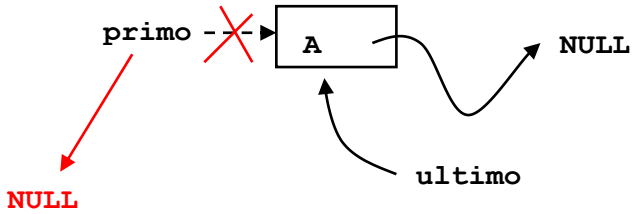


ATTENZIONE! Se l'elemento è l'unico in coda bisogna reimpostare anche il puntatore all'ultimo elemento:

1

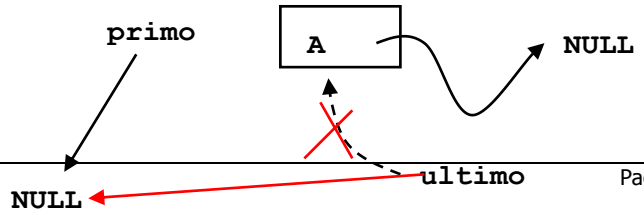


2




```
coda.primo = NULL;
```

3




```
coda.ultimo = NULL;
```