CORSO DI PROGRAMMAZIONE
A.A. 2023-24

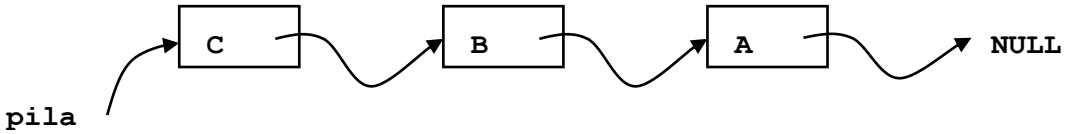
# Dispensa 23

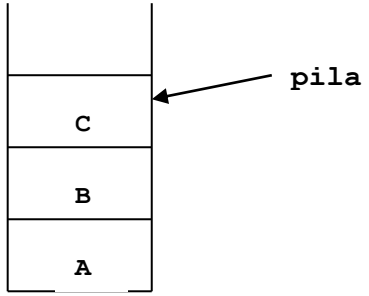
Laboratorio

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

# 23 Le Pile

Una pila è una lista con un puntatore alla testa dell'elenco delle celle, con l'obbligo di eseguire tutte le operazioni a partire dalla testa:




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

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

struct cella *pila = NULL; //puntatore alla testa
```

In una pila tutti i nuovi elementi vengono inseriti in testa, tutte le altre operazioni vengono fatte sempre a partire dalla testa, ma ogni volta che si va a considerare un elemento questo viene tolto dalla pila.

Solitamente per operare con le pile si utilizzano due funzioni chiamate "*push()*" e "*pop()*", rispettivamente per l'inserimento e la lettura all'interno di una pila. Attraverso queste due funzioni si eseguiranno tutte le operazioni sulla pila. La pila è una struttura dati di tipo LIFO (Last In First Out) dove l'ultimo elemento ad essere inserito è il primo ad essere gestito (letto).

# 23.1 Funzione *push()*

La funzione *push()* serve per inserire un elemento all'interno della pila:

```
void push(struct cella **p, int num);
```

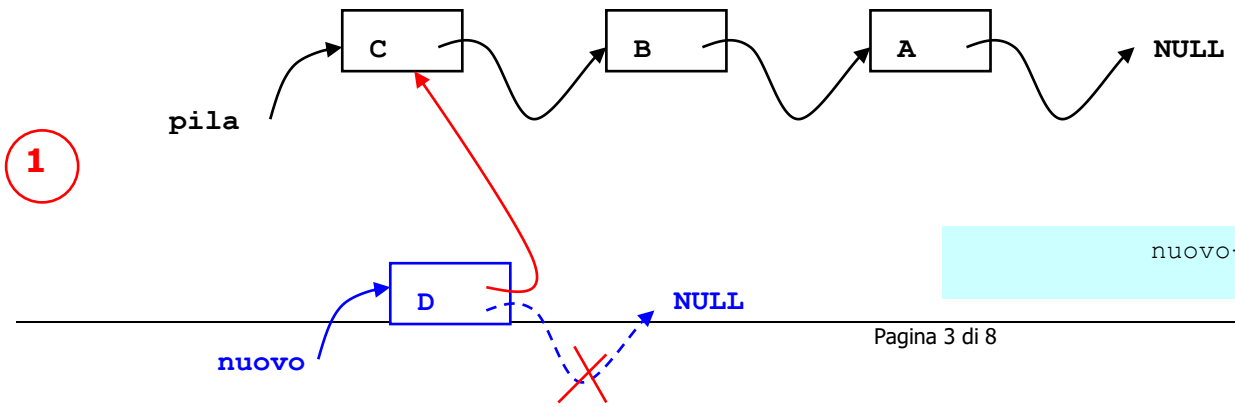
dove "p" è il riferimento alla pila da gestire e "num" è il numero da associare al valore del nuovo elemento da inserire. La funzione *push()*:

```
void push(struct cella **p, int num)
{
      struct cella *nuovo;
      nuovo = (struct cella*)malloc(sizeof(struct cella));

      if (nuovo == NULL)
      {
            printf("\n\n Errore: Il programma verra\' chiuso!\n");
            system("pause");
            exit(1);
      }
      nuovo->valore = num;
      nuovo->next = *p;
      *p = nuovo;
}
```

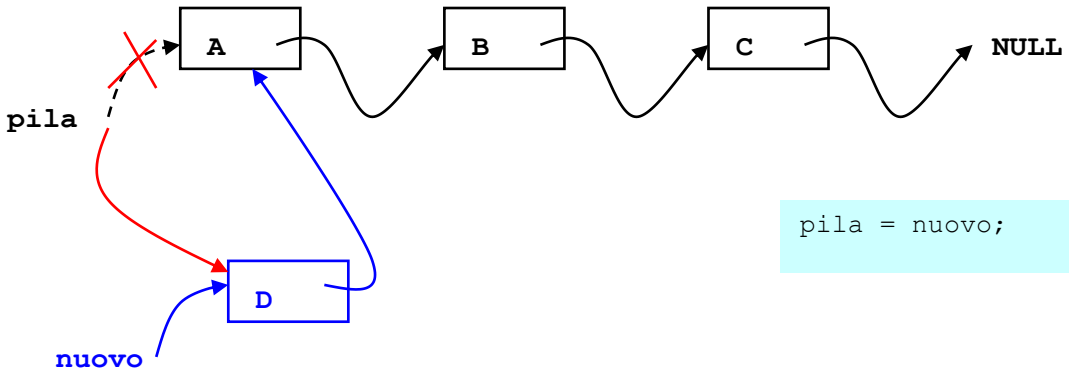
Da notare che la funzione *push()* corrisponde ad un semplice inserimento in testa ad una lista semplice.

La prima operazione da compiere è collegare la nuova cella alla prima della lista, quindi fare in modo che l'elemento "next" della nuova cella prenda lo stesso valore di "pila":



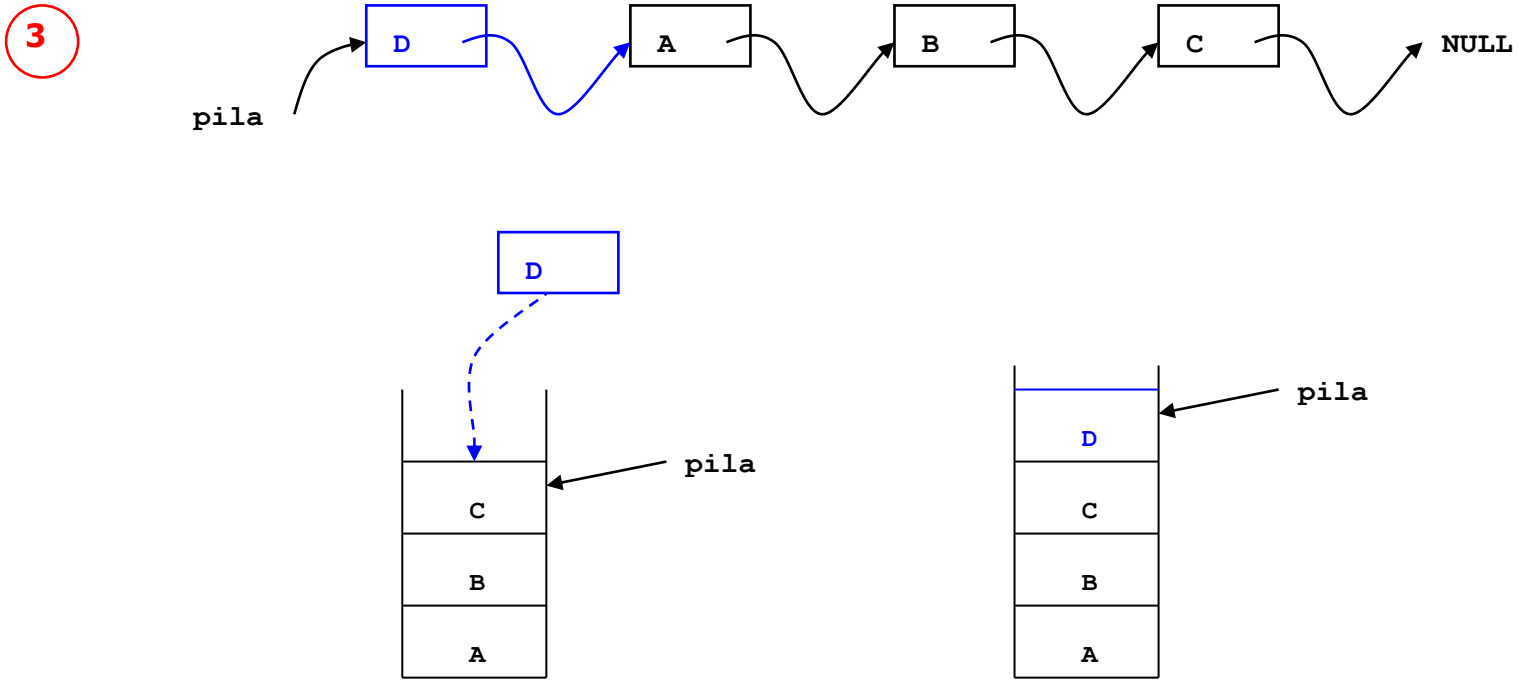

```
nuovo->next = pila;
```

La seconda operazione consiste nell'associare la testa al nuovo elemento creato:




```
pila = nuovo;
```

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



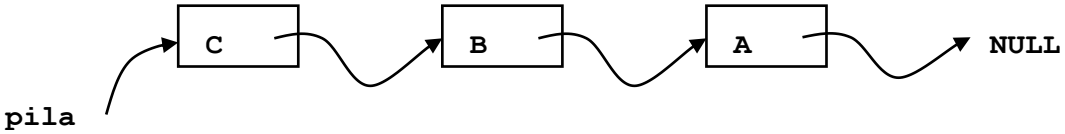
## 23.2 Funzione *pop()*

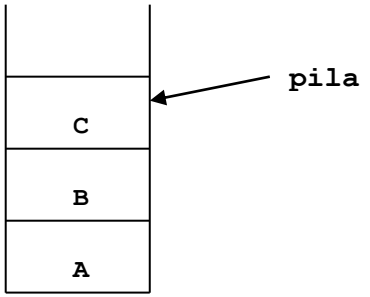
La funzione *pop()* consente di leggere il primo elemento in testa alla pila restituendo un puntatore. Notare che tale elemento viene poi eliminato dalla pila:

```
struct cella *pop(struct cella **p);
```

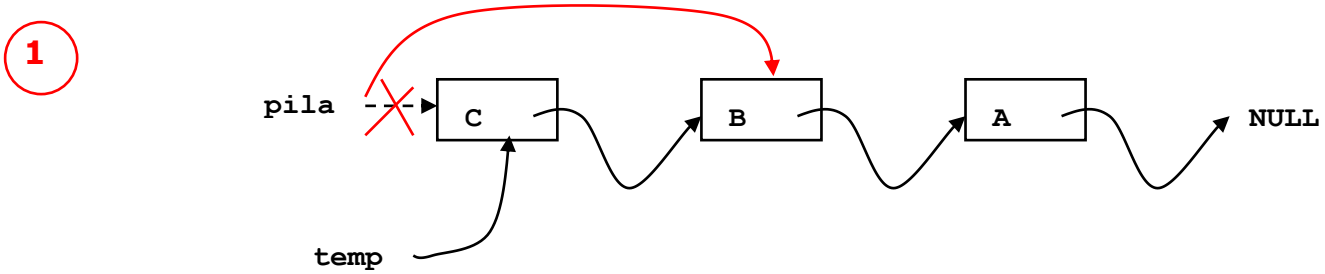
dove "p" è il riferimento alla pila da gestire.

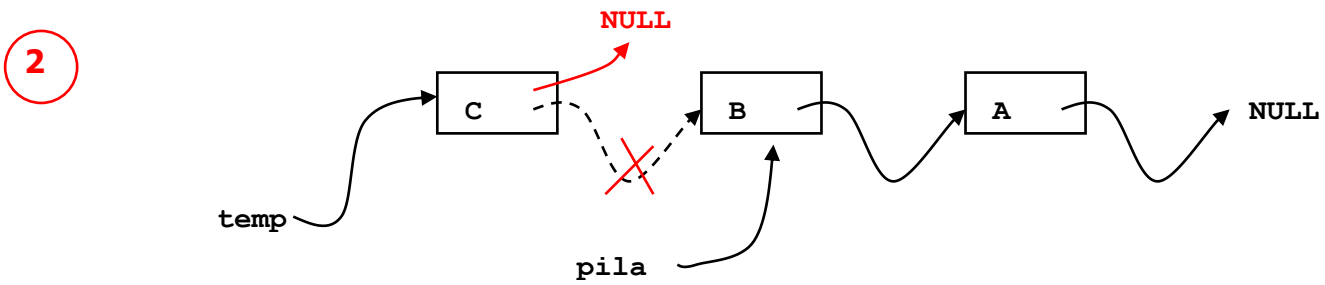
Consideriamo la seguente pila:



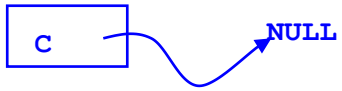



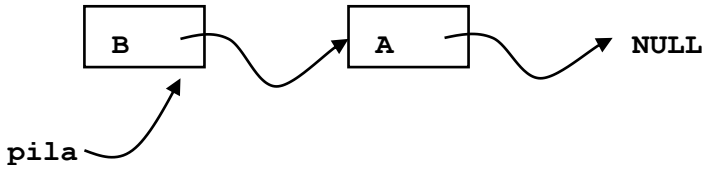

Dopo la chiamata della funzione *pop()* avremo:

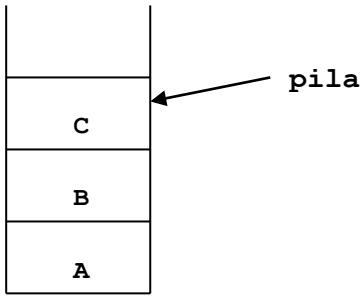


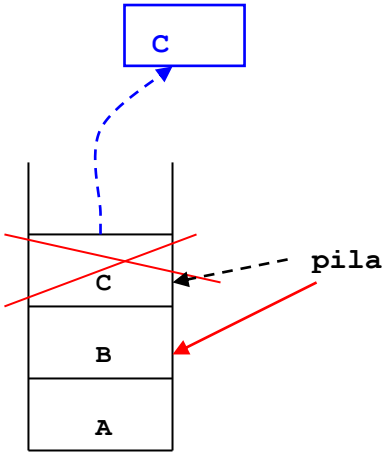



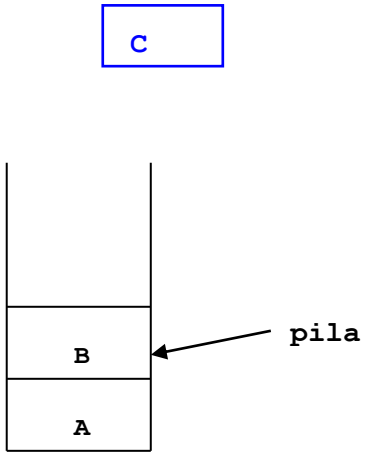
3




Dove in questo caso la cella "C" è quella restituita dalla funzione!

La funzione *pop()*:

```
struct cella* pop(struct cella **p)
{
      struct cella *temp;
      if (*p == NULL)
           return *p;

      temp = *p;
      *p = (*p)->next;
      temp->next = NULL;
      return temp;
}
```

**Esercizio 1**

Creare un programma per la gestione di una struttura Pila che chieda all'utente di digitare:

1 per "Inserisci nuovo elemento"

2 per "Stampa elenco elementi"

3 per "Azzera la pila"

4 per "Esci"

Nel menu switch predisporre una chiamata ad una funzione per ciascun valore da 1 a 3. Usare i seguenti prototipi di funzioni:

```
void push(struct cella **t, int num);
void stampaPila(struct cella **t);
void resetPila(struct cella **t);
```

E all'interno della funzione *stampaPila* e *resetPila* chiamare la funzione *pop* con il seguente prototipo:

```
struct cella* pop(struct cella **t);
```

Gestire le chiamate sopra elencate partendo da una Pila vuota. Usare questa struttura per la cella:

```
struct cella
{
   int ordine;
   struct cella *next;
};
struct cella *testa = NULL;
```