UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Programmazione II e Laboratorio di PII

Pila e Coda

Angelo Ciaramella

Pila e Coda

- La pila (o stack) e la coda (o queueue) sono casi speciali di un tipo di dati più generale, le liste ordinate

$$A = a_0, a_1, \ldots, a_{n-1} \qquad n \geq 0$$

- Il termine a_0 rappresenta il generico elemento (o atomo) che è estratto da un insieme

Pila di dati

- Una Pila è una lista ordinata dove gli inserimenti e le cancellazioni vengono effettuati ad un estremo della lista chiamata top

- Dato uno stack

$$S = a_0, a_1, \ldots, a_{n-1}$$

- a_0 rappresenta l'elemento di base (bottom element), a_{n-1} è l'elemento in cima allo stack (top element)

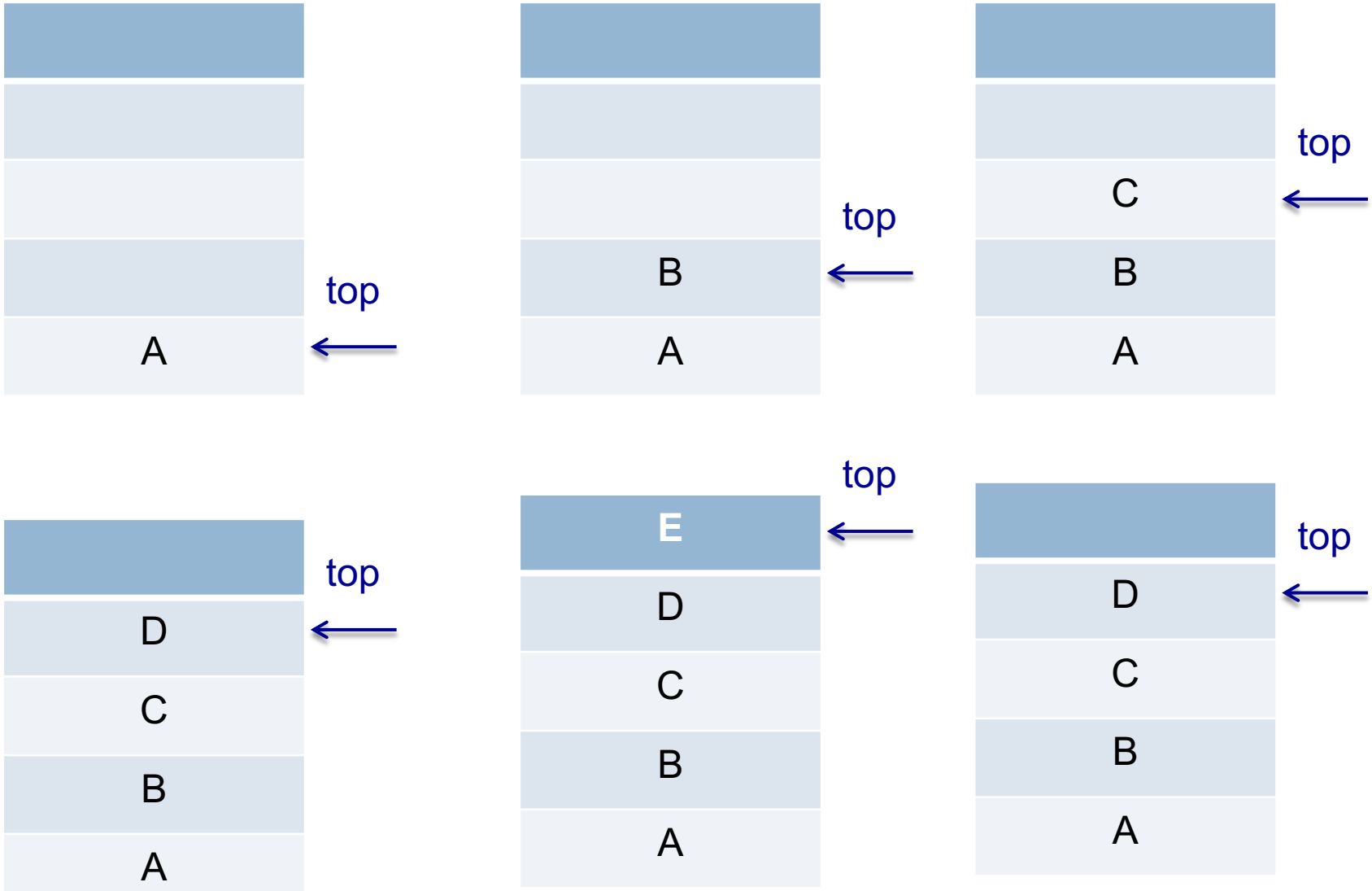
Pila di dati

- L'ultimo elemento che viene inserito in uno stack è il primo ad essere eliminato
  - Lista LIFO (Last-In-First-Out)
- L'operazione di inserimento è detta Push e l'operazione di cancellazione è detta Pop
- Uno stack speciale è lo stack di sistema
  - Utilizzato dai programmi per gestire ed elaborare le chiamate delle funzioni

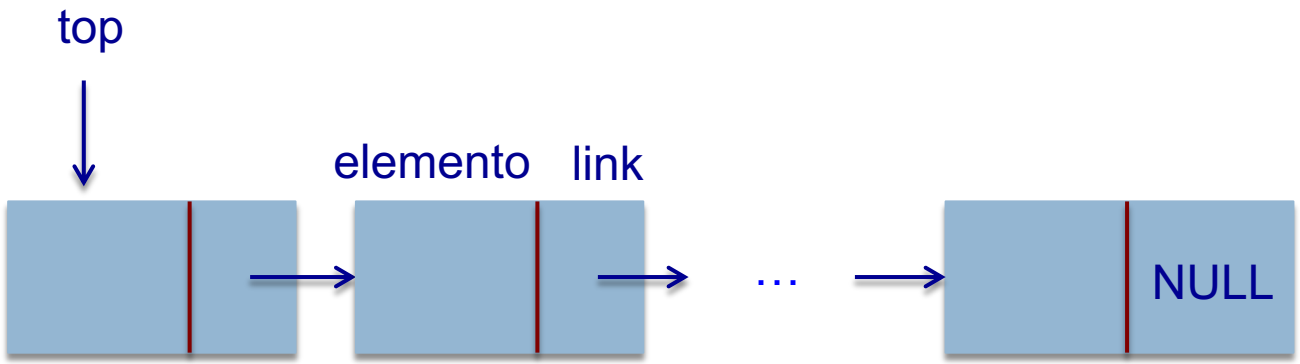
Tipo dato Pila

```
tipo Pila:

dati:
  una sequenza S di n elementi

operazioni:
    isEmpty() → result
      restituisce true se S è vuota, e false
      altrimenti

    push(elem e)
      aggiunge e come ultimo elemento di S

    pop() → elem
      toglie da S l'ultimo elemento e lo
      restituisce

    top() → elem
      restituisce l'ultimo elemento di S (senza eliminarlo)
```


Operazioni sullo Stack



Pila con lista concatenata



Pila con lista concatenata

Rappresentazione di uno stack

```
typedef struct stack *stack_pointer;

typedef struct stack {
          int item;
          stack_pointer link;
};
```


Operazione di push

```
void push(stack_pointer *top, int item){
/* aggiunge un elemento in cima allo stack */

stack_pointer temp = (stack_pointer) malloc (sizeof
(stack));

/* controllo IS_FULL(temp) */

temp->item = item;
temp->link = *top;
*top = temp;

}
```


Operazione di pop

```
int pop(stack_pointer *top){
/* cancella un elemento in cima allo stack */
int item;

stack_pointer temp = *top;

/* controllo IS_FULL(temp) */

item = temp->item;
*top = temp->link;

free(temp);

return item;
}
```


Coda

- Una coda (queue) è una lista ordinata nella quale tutti gli inserimenti avvengono ad un estremo e tutte le cancellazioni avvengono all'estremo opposto

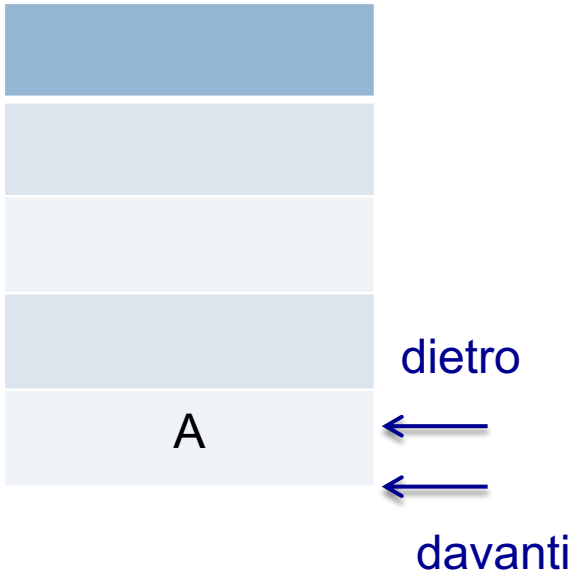
$$Q = a_0, a_1, \ldots, a_{n-1}$$

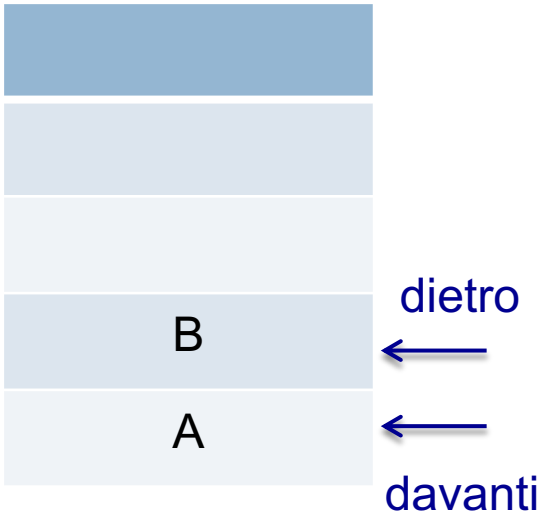
- Il primo elemento che viene inserito è il primo ad essere eliminato
  - Lista FIFO (First-In-First-Out)

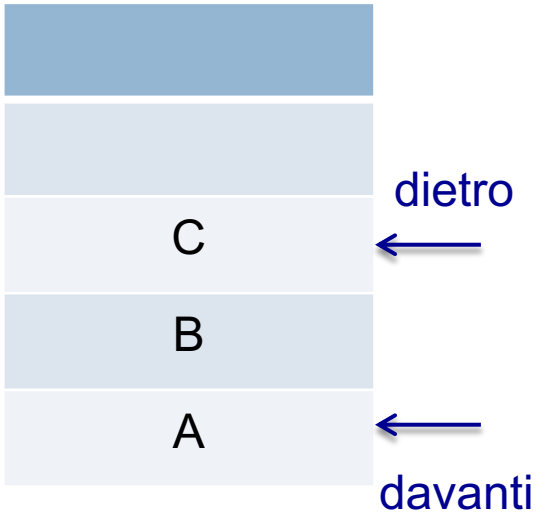
Tipo dato Coda

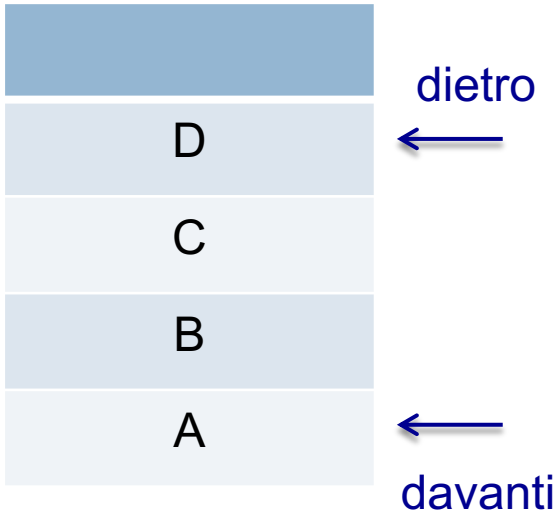
```
tipo Coda:

dati:
  una sequenza S di n elementi

operazioni:
    isEmpty()→ result
      restituisce true se S è vuota, e false
      altrimenti

    enque(elem e)
      aggiunge e come ultimo elemento di S

    dequeue()→ elem
      toglie da S il primo elemento e lo restituisce

    top() → elem
      restituisce il primo elemento di S (senza eliminarlo)
```

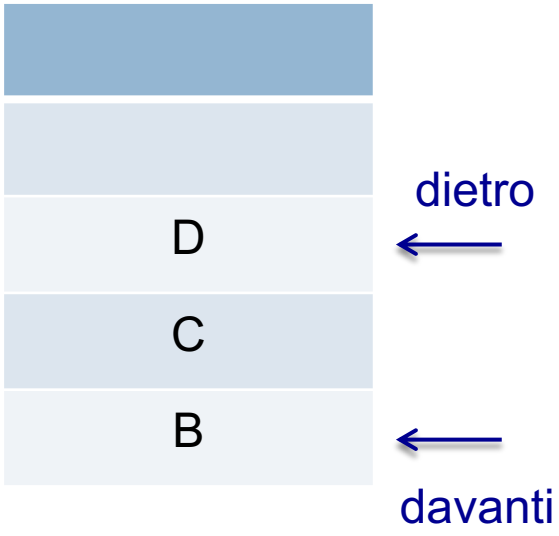

Operazioni sulla coda



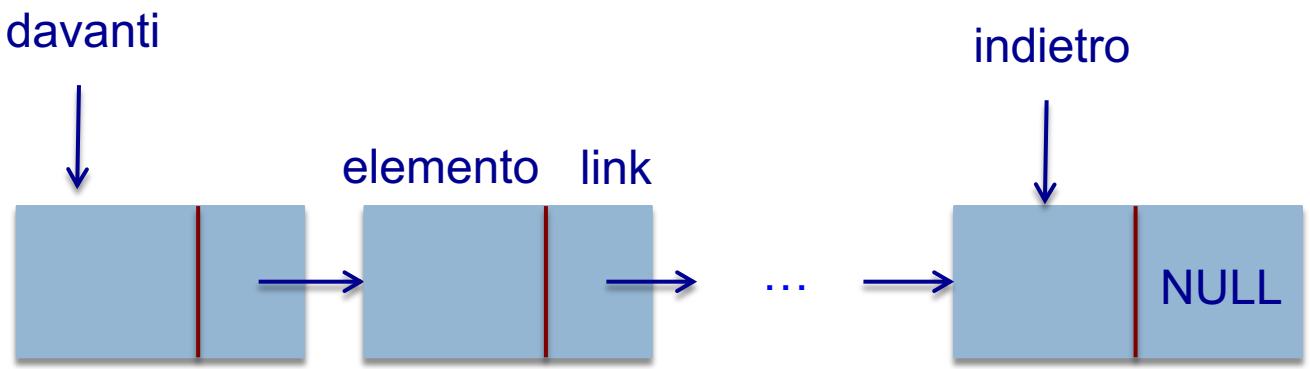








Coda concatenata



Coda con lista concatenata

Rappresentazione di una coda

```
typedef struct coda *coda_pointer;

typedef struct coda {
          int item;
          coda_pointer link;
};
```


Operazione di enqueue

```
void enqueue(coda_pointer *davanti, coda_pointer
*dietro, int item){
/* aggiunge un elemento in fondo alla coda */

coda_pointer temp = (coda_pointer) malloc (sizeof
(coda));

/* controllo IS_FULL(temp) */

temp->item = item;
temp->link = NULL;

if (*davanti) (*dietro)->link = temp;
else *davanti = temp;
*dietro = temp;

}
```


Operazione di dequeue

```
int dequeue(coda_pointer *davanti){
/* cancella il primo elemento di una coda */

int item;
coda_pointer temp = *davanti;

/* controllo IS_FULL(temp) */

item = temp->item;
*davanti = temp->link;

free(temp);

return item;
}
```