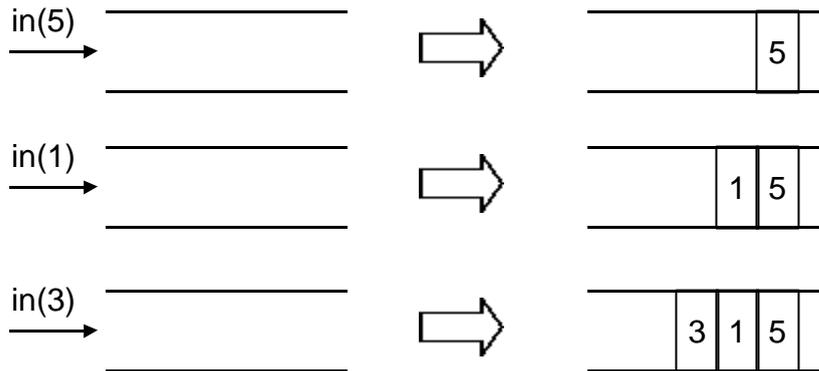


QUEUE (CODE)

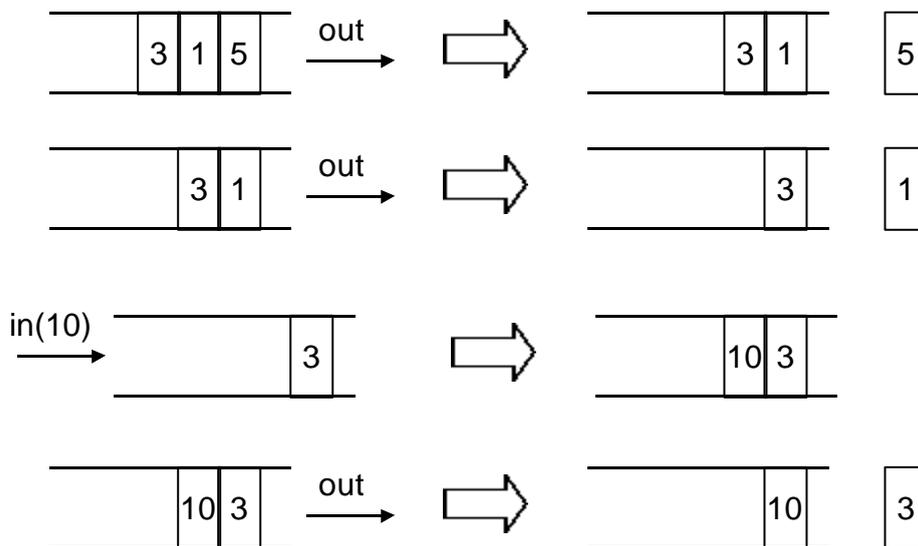
Una **coda (queue)** e' una struttura che memorizza dati omogenei, permettendone l'inserimento e l'estrazione di essi in modalita' **FIFO** (first-in-first-out)



1

Corrado Santoro - Laboratorio di Informatica - Lezione 7 - CdS Ing. Informatica - Universita' di Catania

QUEUE (CODE)



2

Corrado Santoro - Laboratorio di Informatica - Lezione 7 - CdS Ing. Informatica - Universita' di Catania

QUEUE (CODE)

↑ fine (tail) ↑ testa (head)

Su una coda vengono definite due funzioni:

| | |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <p>IN (elemento) <i>enqueue (elemento)</i> <i>put (elemento)</i></p> | <p>inserisce l'elemento alla fine della coda (se questa non e' pieno)</p> |
| <p>OUT () → elemento <i>dequeue ()</i> <i>get ()</i></p> | <p>Estrae l'elemento in testa alla coda (se questa non e' vuoto), e lo restituisce</p> |

3

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

Usiamo un vettore statico e definiamo la seguente struttura dati:

```

#define QUEUE_SIZE ...   Dimensione massima della coda

typedef struct {
    int head;             Indice dell'elemento in testa
    int tail;             Indice dell'elemento finale
    int data [QUEUE_SIZE]; Array che contiene i dati della coda
} t_queue;
        
```

5 4 3 2 1 0

↑ h = 0
↑ t = 0

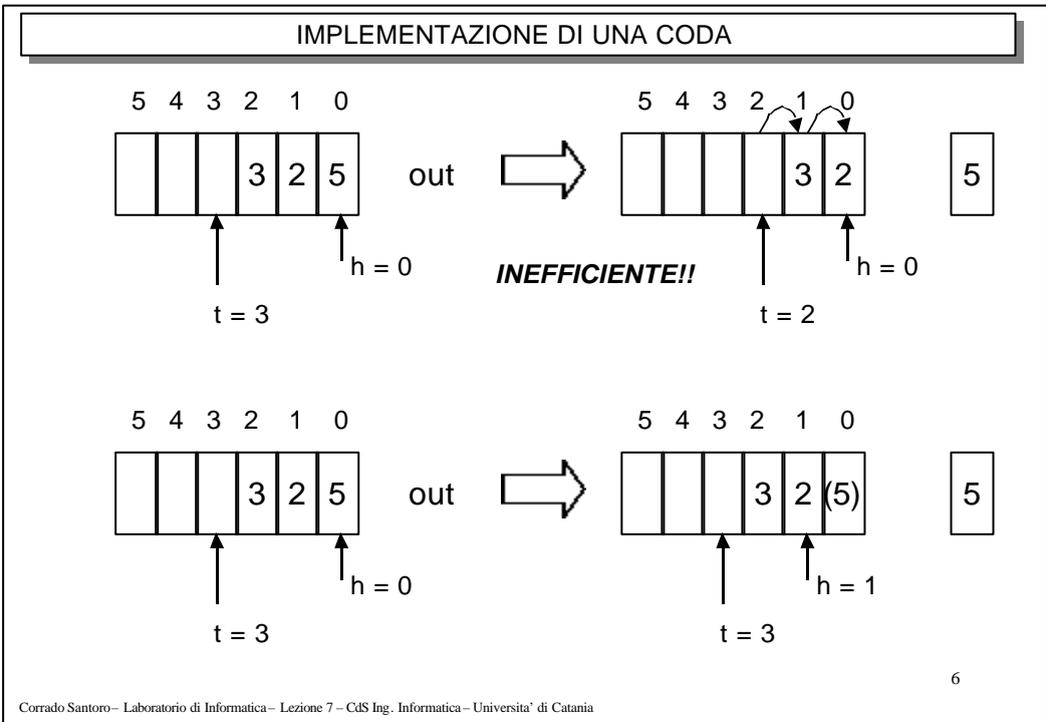
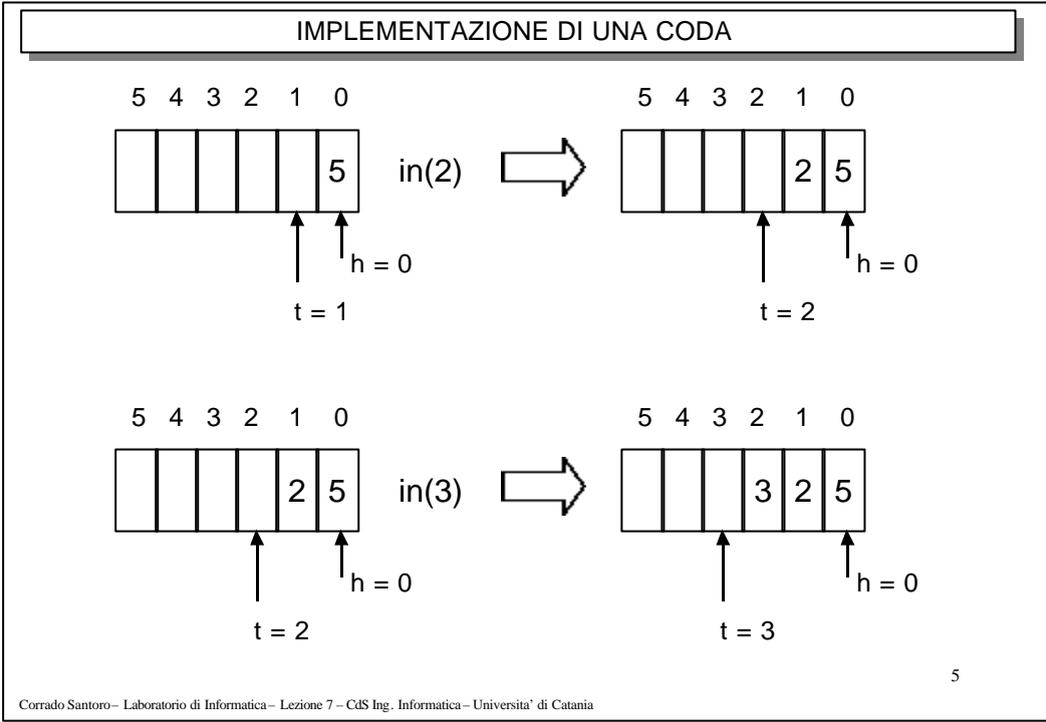
in(5) →

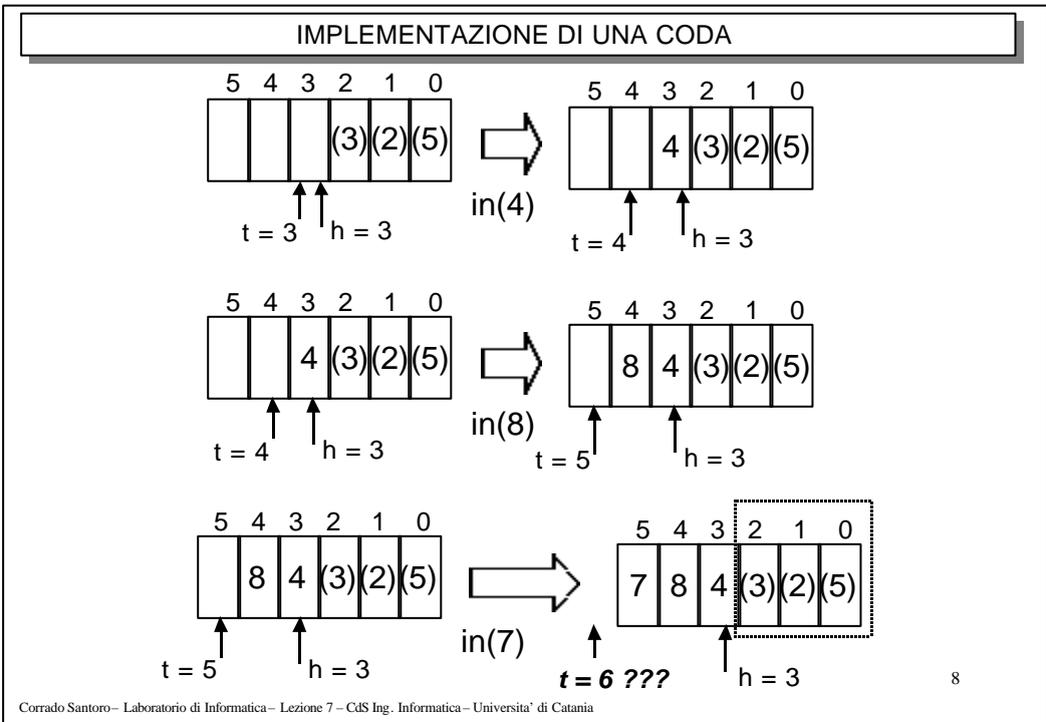
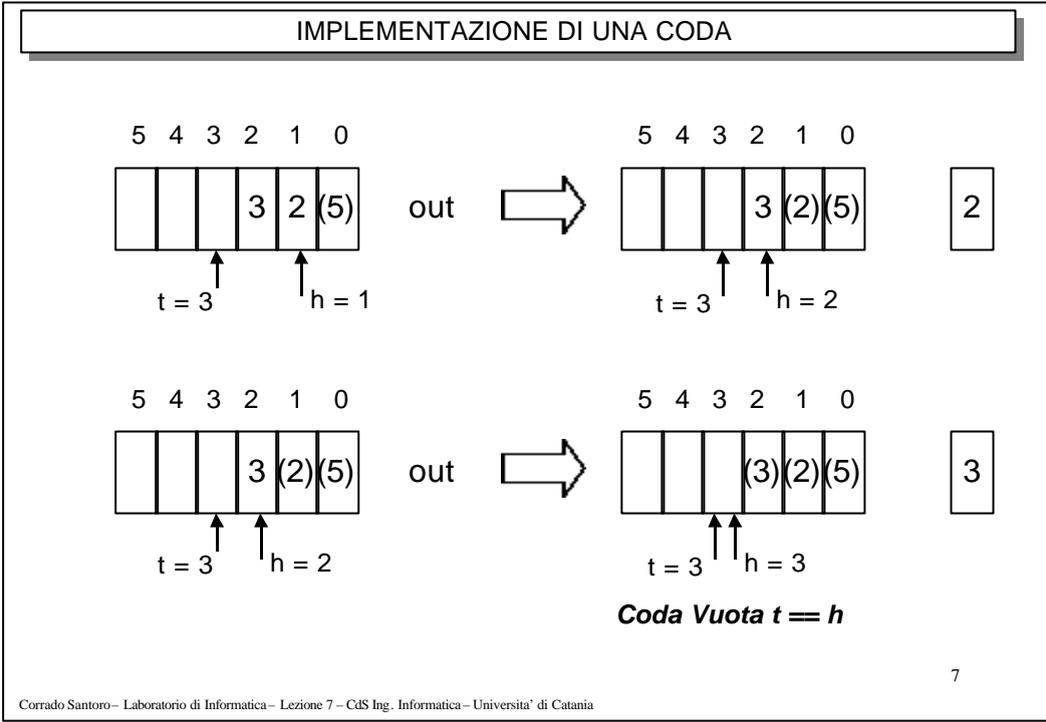
5 4 3 2 1 0

↑ h = 0
↑ t = 1

4

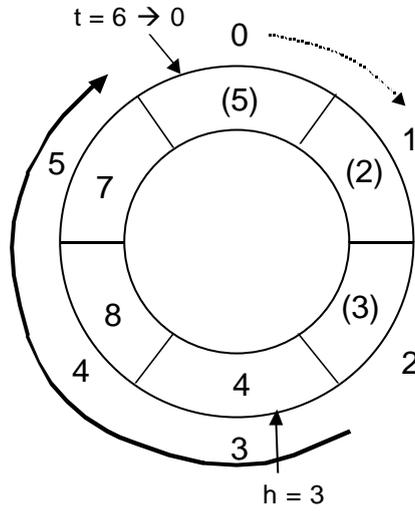
Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania





CODA CIRCOLARE

Disegniamo il vettore come una *circonferenza*, unendo i due estremi



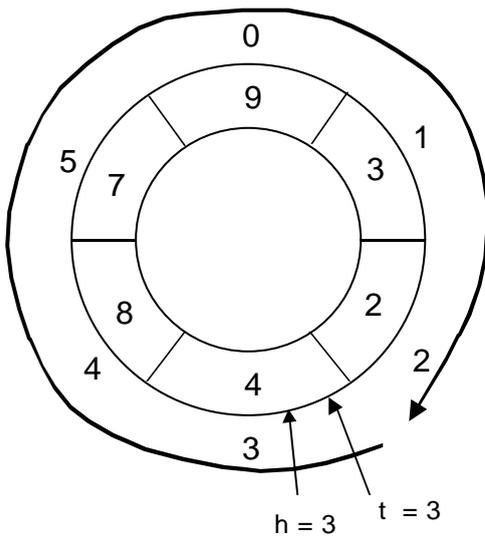
$$t = (t + 1) \% \text{QUEUE_SIZE}$$

$$h = (h + 1) \% \text{QUEUE_SIZE}$$

9

CODA CIRCOLARE

in(9), in(3), in(2)



Coda Piena $t = h$

Stessa condizione di coda vuota????

10

CODA CIRCOLARE – CONDIZIONI DI PIENO E VUOTO

```
#define QUEUE_SIZE ... Dimensione massima della coda

typedef struct {
    int head; Indice dell'elemento in testa
    int tail; Indice dell'elemento finale
    int n; Numero di elementi effettivamente presenti in coda
    int data [QUEUE_SIZE]; Array che contiene i dati della coda
} t_queue;
```

IN (elemento) ***n ++***

OUT () → elemento ***n --***

coda piena ***n == QUEUE_SIZE***

coda vuota ***n == 0***

11

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

```
#define QUEUE_SIZE ...

typedef struct {
    int head;
    int tail;
    int n;
    int data [QUEUE_SIZE];
} t_queue;

void in (t_queue * q, int value)
{
    q->data [q->tail] = value;
    q->tail = (q->tail + 1) % QUEUE_SIZE;
    q->n ++;
}
```

12

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

```
#define QUEUE_SIZE ...

typedef struct {
    int head;
    int tail;
    int n;
    int data [QUEUE_SIZE];
} t_queue;

int out (t_queue * q)
{
    int ret_value;
    ret_value = q->data [q->head];
    q->head = (q->head + 1) % QUEUE_SIZE;
    q->n --;
    return ret_value;
}
```

13

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

```
t_queue init_queue (void)
{
    t_queue new_queue;
    new_queue.head = 0;
    new_queue.tail = 0;
    new_queue.n = 0;
    return new_queue;
}

int is_queue_empty (t_ queue q)
{
    return q.n == 0;
}

int is_queue_full (t_ queue q)
{
    return q.n == QUEUE_SIZE;
}
```

14

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

```
typedef struct {
    int head;
    int tail;
    int n;
    int data [QUEUE_SIZE];
} t_queue;

void dump_queue (t_queue q)
{
    int i, index;
    index = q.head;
    for (i = 0; i < q.n; i++) {
        printf ("%d ", q.data [index]);
        index = (index + 1) % QUEUE_SIZE;
    }
    printf ("\n");
}
```

15

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNA CODA

```
int main (int argc, char * argv[])
{
    t_queue myqueue;
    int choice;

    myqueue = init_queue ();
    do {
        printf ("0. Exit\n");
        printf ("1. Inserisci elemento\n");
        printf ("2. Estrai elemento\n");
        printf ("3. Stampa elemento\n");
        scanf ("%d", &choice);
    }
    ...
}
```

16

Corrado Santoro – Laboratorio di Informatica – Lezione 7 – CdS Ing. Informatica – Università di Catania

IMPLEMENTAZIONE DI UNO STACK

```
...
switch (choice) {
  case 1:
  {
    int elem;
    printf ("Inserisci l'elemento :");
    scanf ("%d", &elem);
    if (is_queue_full (myqueue) == 0)
      in (&myqueue, elem);
    else
      printf ("Coda piena, inserimento fallito\n");
  }
  break;
...

```

17

Corrado Santoro - Laboratorio di Informatica - Lezione 7 - CdS Ing. Informatica - Università di Catania

IMPLEMENTAZIONE DI UNO STACK

```
...
  case 2:
  {
    int elem;
    if (is_queue_empty (myqueue) == 0) {
      elem = out (&myqueue);
      printf ("Estrazione riuscito = %d\n", elem);
    }
    else
      printf ("Coda vuoto, out fallito\n");
  }
  break;
  case 3:
    dump_queue (myqueue);
    break;
}
} while (choice != 0);
}

```

18

Corrado Santoro - Laboratorio di Informatica - Lezione 7 - CdS Ing. Informatica - Università di Catania