Coda

Una **coda** è un tipo astratto che consente di rappresentare una sequenza di elementi in cui è possibile aggiungere elementi ad un estremo, detto **fondo**, e togliere elementi dall'altro estremo, detto **testa**. Tale strategia di accesso è detta **FIFO**, First In First Out, ed è particolarmente adatta a rappresentare sequenze nelle quali l'elemento viene elaborato secondo l'ordine di arrivo (ad esempio le liste d'attesa).

Specifica sintattica

Tipi: `coda, boolean, tipoelem`
Operatori:

- `creacoda: () → coda`
- `codavuota: (coda) → boolean`
- `leggicoda: (coda) → tipoelem`
- `fuoricoda: (coda) → coda`
- `incoda: (tipoelem, coda) → coda`

Specifica semantica

Tipi:

- `coda`: insieme delle sequenze $q = a_1, a_2, \ldots, a_n$ di elementi di tipo `tipoelem` con accesso FIFO
- `boolean`: insieme dei valori di verità

Operatori:

- `creacoda = q'`
  - Post: $q' = <>$
- `codavuota(q) = b`
  - Post: $b = true$ se $q = <>$
    $b = false$ altrimenti
- `leggicoda(q) = a`
  - Pre: $q = <a_1, a_2, \ldots, a_n>$ con $n \geq 1$
  - Post: $a = a_1$
- `fuoricoda(q) = q'`

- Pre: $q = < a_1, a_2, \ldots, a_n >$ con $n \geq 1$
- Post: $q' = < a_2, a_3, \ldots, a_n >$ con $n > 1$

  $q' = < >$ con $n = 1$

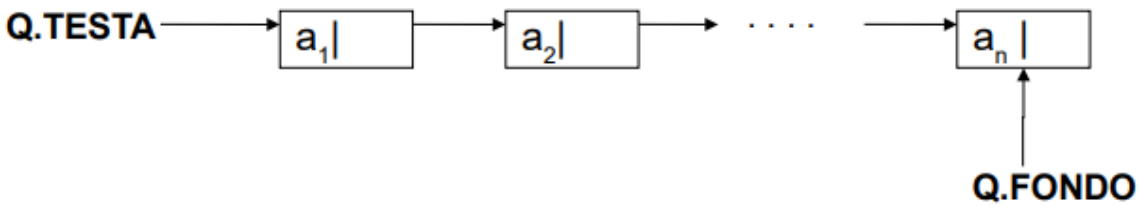
- `incoda(a, q) = q'`
  - Pre: $q = < a_1, a_2, \ldots, a_n >$ con 0
  - Post: $q' = < a_1, a_2, \ldots, a_n, a >$

Rappresentazione

In generale le possibili rappresentazioni delle code sono **analoghe a quelle delle pile** con l'attenzione che è conveniente consentire l'accesso sia all'elemento inserito per primo sia all'elemento inserito per ultimo.

Realizzazione con puntatori

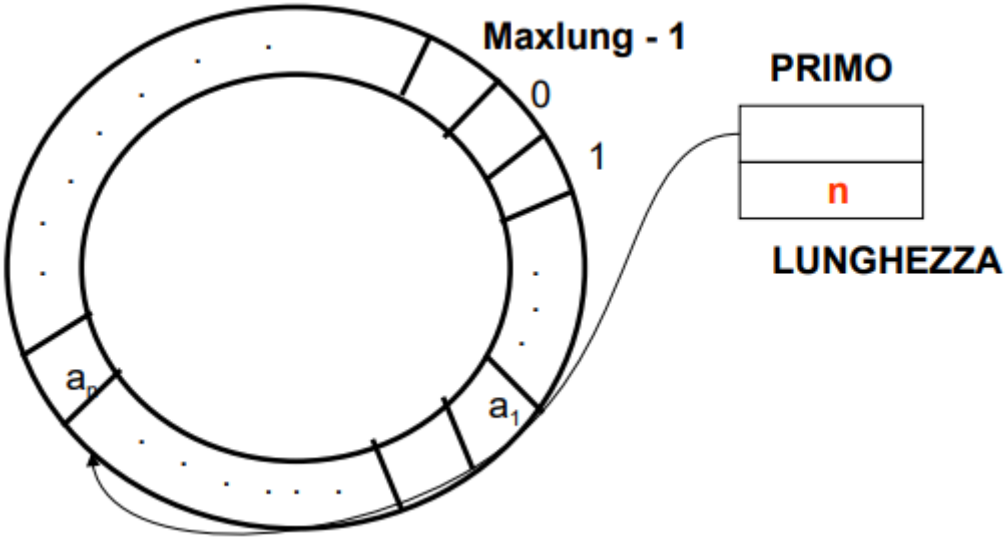
La coda è realizzata con n **celle**, la prima delle quali è indirizzata da un puntatore "**testa**" e l'ultima da un puntatore "**fondo**". La coda vuota è individuata dal valore nullo `null` del puntatore di testa.



Realizzazione con vettore circolare

Per le code la rappresentazione sequenziale non è agevole come per le pile; è utile gestire l'array in modo **circolare**.
Il vettore circolare è inteso come un array di `maxlung` elementi, con indice da 0 a `maxlung-1`, in cui consideriamo l'elemento di indice 0 come successore di quello di indice `maxlung-1`. Si utilizzano due **variabili**:

1. Il valore di **primo** indica la posizione dell'array in cui è memorizzato l'elemento inserito per primo;
2. **Ultimo** si riferisce all'ultimo elemento inserito, oppure definisce la lunghezza della coda.



Esercizi su pile e code

Si vuole realizzare un programma che prende una coda di interi e restituisce un'altra coda ottenuta dalla prima considerando solo valori positivi.
Trascuriamo le dichiarative relative alla implementazione della coda.

```
estrai(coda q, coda q1 per riferimento)
creacoda(q1)
while not codavuota(q) do
      e = leggicoda(q)
      if e > 0 then
            incoda(e, q1)
      fuoricoda(q)
```

Se volessimo conservare la coda originale dovremmo usare una coda ausiliaria.

```
estrai1(coda q per riferimento, coda q1 per riferimento)
creacoda(q1)
creacoda(qaux)
while not codavuota(q) do
      e = leggicoda(q)
      if e > 0 then
            incoda(e, q1)
      fuoricoda(q)
      incoda(e, qaux)
creacoda(q)
while not codavuota(qaux) do
      e = leggicoda(qaux)
      incoda(e, q)
      fuoricoda(qaux)
```