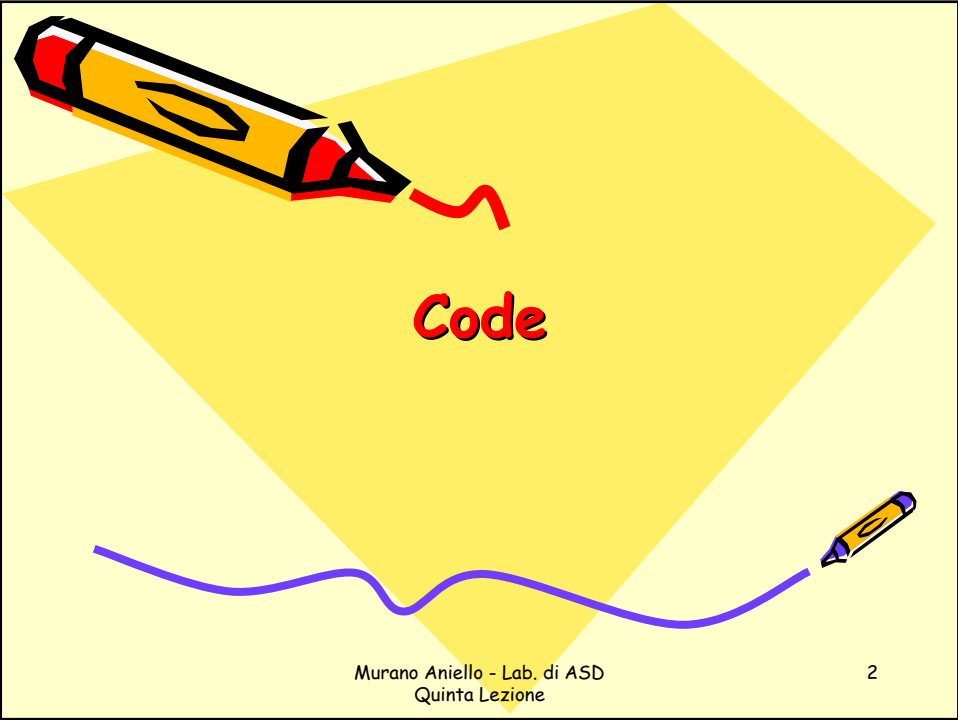


**Laboratorio di Algoritmi e
Strutture Dati**

Aniello Murano
<http://people.na.infn.it/~murano/>

Murano Aniello - Lab. di ASD
Quinta Lezione

1



Code

Murano Aniello - Lab. di ASD
Quinta Lezione

2

Stack(pile) e Code

- Stack e code sono insiemi dinamici in cui l'elemento rimosso dall'insieme con l'operazione di cancellazione è sempre predeterminato.
- In uno stack, l'elemento cancellato è quello più recentemente inserito. Gli stack rispettano la politica LIFO (*last-in, first-out*).
- In una coda, l'elemento cancellato è sempre quello che è rimasto più a lungo nell'insieme. Le code rispettano la politica FIFO (*first-in, first out*).
- In uno stack un nuovo elemento è sempre posto in testa agli altri, mentre nella coda esso è posto dopo tutti gli altri.
- Ci sono molti modi per implementare stack e code su un computer. Cominciamo con una implementazione di stack utilizzando array.



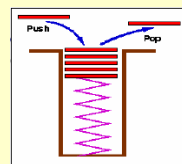
Murano Aniello - Lab. di ASD
Quinta Lezione

3

Stack

- Le operazioni fondamentali su Stack sono:
- $Push(S,x)$ che serve a inserire l'elemento x al top dello stack
- $Pop(S)$ che serve a cancellare il top dello stack S

- Un esempio di stack può essere dato da una pila di piatti posta su un tavolo.



- Si noti che l'ordine in cui i piatti sono tolti dallo stack è inversa all'ordine in cui essi sono stati inseriti nello stack, visto che solo il top dello stack è accessibile.
- Un'altra operazione importante sugli stack $Empty-Stack$, necessaria per controllare se uno stack è vuoto



Murano Aniello - Lab. di ASD
Quinta Lezione

4

Code

- A differenza dello stack, una coda usa due attributi:
 - Inizio coda (Head)
 - Fine coda (Tail)
- In pratica, usando come esempio di coda una fila ad uno sportello. L'inizio della coda è rappresentato dalla prima persona della fila (quella la prossima ad essere servita), mentre la fine della coda è rappresentata dall'ultima persona che si è aggiunta alla coda (cioè, l'ultima persona tra quelle attualmente in fila ad essere servita)
- Un inserimento nella coda sarà fatto sempre alla sua fine mentre una cancellazione alla sua testa

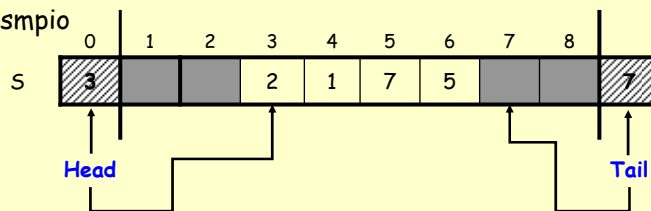


Murano Aniello - Lab. di ASD
Quinta Lezione

5

Implementazione di Code

- Come per gli stack, anche per le code possiamo avere diversi modi di rappresentazione su un computer.
- Iniziamo con l'utilizzo di array.
- Per memorizzare una coda con massimo n elementi, possiamo utilizzare uno stack di dimensione n e di due variabili che memorizzano costantemente l'indice della testa e della coda.
- Come per gli stack, si può anche scegliere di memorizzare l'indice della testa e della coda nel vettore stesso.
- Per esempio



Murano Aniello - Lab. di ASD
Quinta Lezione

6

Implementazione di Dequeue e Enqueue

- Usando l'array Q precedente per l'implementazione di una coda, le operazioni di Cancellazione (Dequeue) e Inserimento (Enqueue) possono essere realizzate come segue:

Enqueue

```
void Enqueue(int Q[], int valore) {
    Q[Q[MAX+1]] = valore;
    if (Q[0]==0) Q[0]=1;
    if (Q[MAX+1] == MAX)
        Q[MAX+1]=1;
    else Q[MAX+1]= Q[MAX+1]+1; }
```

Dequeue

```
int Dequeue(int Q[]) {
    int x=Q[Q[0]];
    if (Q[0]+1 == Q[MAX+1]) {
        Q[0]=0;
        Q[MAX+1]=1;
    }
    else if (Q[0] == MAX) Q[0] = 1;
    else Q[0] = Q[0] + 1;
    return x;
}
```



Stampa e EmptyQueue

- Questa è funzione per la stampa:

```
void Stampa_Queue(int Q[]) {
    int i=Q[0];
    printf("\n Head: %d, Tail: %d ", Q[0], Q[MAX+1]);
    printf("\n Array attuale: ");
    if (Q[0]==0) printf(" QUEUE EMPTY ");
    else { do
        { printf("\|%d",Q[i]);
          if (i<MAX) i++; else i=1; }
      while(i!=Q[MAX+1]);
    printf("\|"); }
```

- Questa è funzione per il controllo della coda vuota:

```
int EmptyQueue(int Q[]) { return Q[0]==0; }
```



Costruzione di una coda...

```
#include <stdio.h>
#define MAX 20
main() {
int Q[MAX+2],size,i,scelta,valore;
printf("\nQuanti elementi deve contenere inizialmente la coda (max %d
elementi): ", MAX);
scanf("%d",&size);
while (size>MAX)
{printf("\n max %d elementi: ", MAX);
scanf("%d",&size);}
for (i=1;i<=size;i++) {
printf("\nInserire il %d° elemento: ",i);
scanf("%d",&Q[i]); }
if (size>0) Q[0]=1;
if (size==MAX) Q[MAX+1]=1; else Q[MAX+1]=size+1;
.....
```



Murano Aniello - Lab. di ASD
Quinta Lezione

9

...chiamate a Dequeue e Enqueue

```
do {
Stampa_Queue(Q);
printf("\n scelta 1-Dequeue, 2-Enqueue, 3-USCITA : ");
scanf("%d",&scelta);
switch (scelta) {
case 1: if (!EmptyQueue(Q))
printf("\n Ecco il valore in testa %d", Dequeue(Q));
else printf("\n spiacente, coda vuota");
break;
case 2: if (Q[0]!=Q[MAX+1]) {
printf("\n valore da inserire nella coda: ");
scanf("%d",&valore);
enqueue(Q,valore); }
else printf("\n spiacente, coda piena"); } /* fine switch */
} /* fine do */
while(scelta==1||scelta==2);
} /* fine main */
```

Il sorgente è in Queue.c



Murano Aniello - Lab. di ASD
Quinta Lezione

10