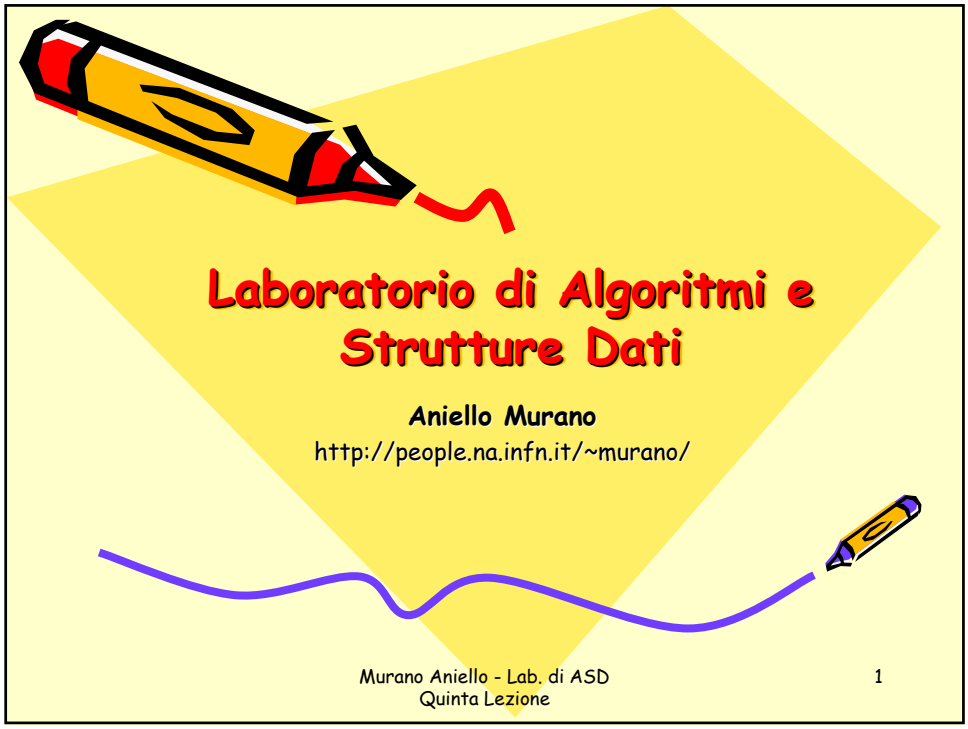
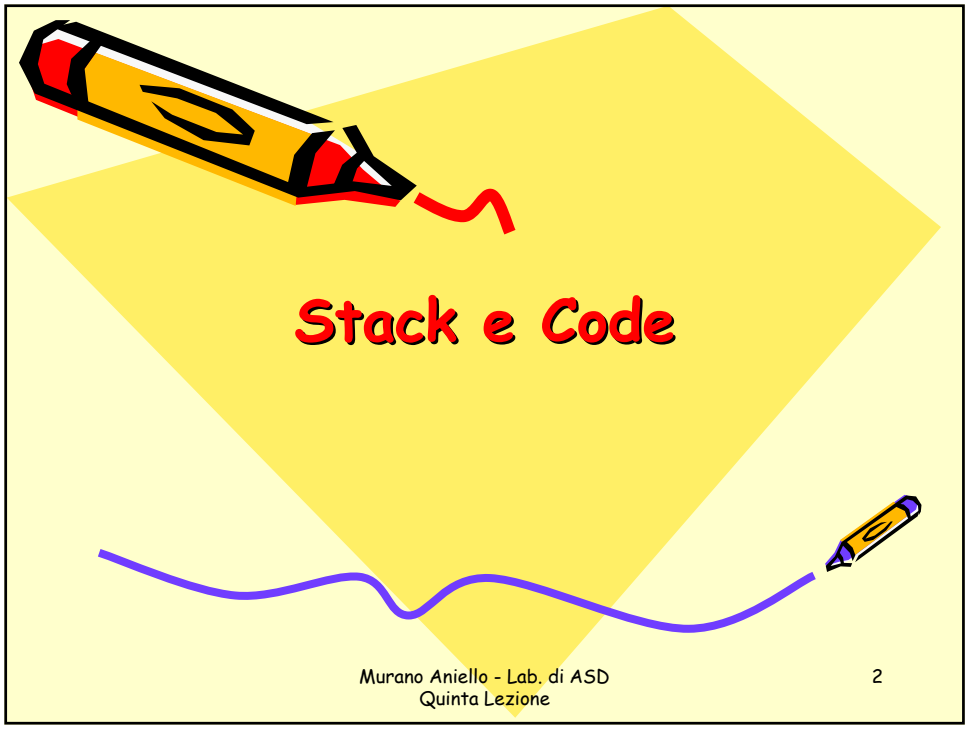


Laboratorio di Algoritmi e Strutture Dati

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



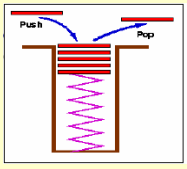
Stack e Code

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.

Stack

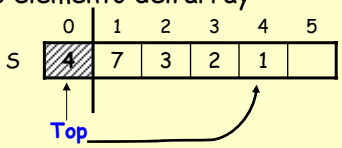
- Le operazioni fondamentali su Stack sono:
- Push(S,x) che serve a inserire l'elemento x al top dello stack
- Pop(S) che serve a cancellare il top dello stack S
- Un esempio di stack può essere dato da una pila di piatti posta su un tavolo.



- 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

Implementazione di Stack con Array

- Per implementare uno stack di n elementi, si può utilizzare un array S di dimensione n.
- L'array S utilizzerà l'attributo TOP che indicherà l'indice dell'elemento più recente immesso nello stack. In una implementazione si potrebbe pensare di memorizzare questo indice nel primo elemento dell'array
- Esempio:



- Un'operazione di Pop sullo stack dell'esempio precedente, restituirà l'elemento 1 e il TOP diventerà 3.
- Quando TOP=0, lo stack è vuoto, cioè non contiene nessun elemento.

Implementazione di Push e Pop

- Utilizzando un array S per l'implementazione di uno stack, nel modo descritto precedentemente, le operazioni di Push, Pop e EmptyStack possono essere realizzate semplicemente:

Push

```
void Push(int S[], int valore)
{
        S[0] = S[0]+ 1;
        S[S[0]]= valore;
}
```

Pop

```
int Pop(int S[])
{
        S[0] = S[0]-1;
        return S[S[0]+1];
}
```

EmptyStack

```
int EmptyStack(int S[])
{
        return S[0]==0;
}
```

Stampa di uno Stack

```
void Stampa_Stack(int S[])
{int i;
  printf("\n Array attuale ");
  for(i=1;i<=S[0];i++) printf("%d",S[i]);
}
```

Costruzione di uno Stack…

```
#include <stdio.h>
#define MAX 20
main()
{
int S[MAX+1],size,i,scelta,valore;
 printf("\nQuanti elementi deve contenere lo Stack (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",&S[i]);}
  S[0]=size;
……..
```


… e chiamata a pop e push

```
do  {
          Stampa_Stack(S);
          printf("\n scelta 1-POP, 2-PUSH, 3-USCITA : ");
          scanf("%d",&scelta);
          switch (scelta) {
                    case 1: if (!EmptyStack(S))
                                  printf("\n Top dello Stack %d", Pop(S));
                              else printf("\n spiacente, stack vuoto");
                              break;
                    case 2: if (S[0]<MAX )      {
                               printf("\n valore da inserire nello stack: ");
                               scanf("%d",&valore);
                               Push(S,valore);  }
                              else printf("\n spiacente, stack pieno");
                                   } /* fine switch */
          }  /* fine do */
   while(scelta==1||scelta==2);
}   /* fine main() */
```

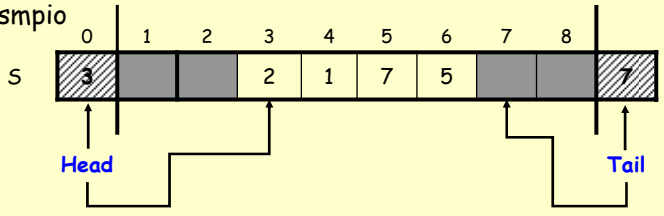
Il sorgente è in Stack.c

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

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 esmpio



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, Teal: %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;
……
```


…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