Code

Pile e Code

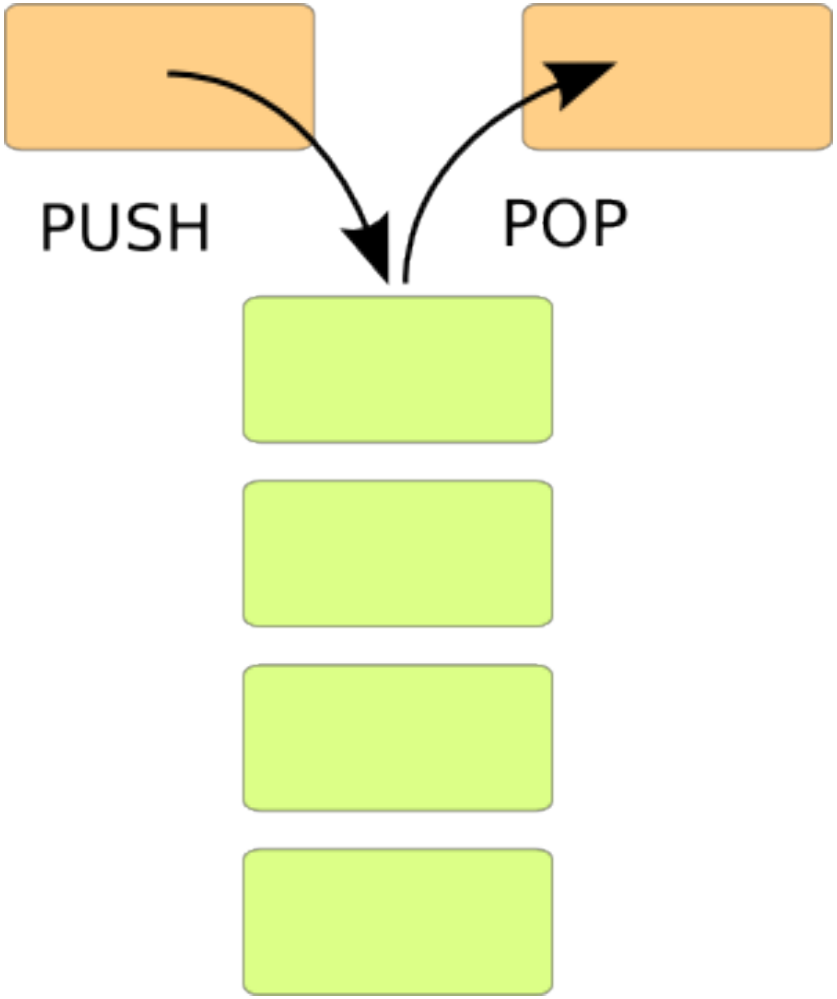
- Pile (Stack) e Code (Queue) sono insiemi dinamici
- Sono definite le operazioni di inserzione e cancellazione
- L'operazione di cancellazione elimina un elemento predefinito
- Pile e code si distinguono per le politiche di eliminazione

Pila (Stack)

- Il termine Pila (Stack) ed il nome delle operazioni di inserzione (push) e cancellazione (pop) derivano da quelli usati per descrivere le operazioni di caricamento per le *pile di piatti a molla*:
  - solo il piatto in cima alla pila può essere rimosso.
  - nuovi piatti vengono aggiunti sempre in cima alla pila

- La politica di cancellazione è del tipo *ultimo arrivato primo servito* o LIFO (Last In First Out)

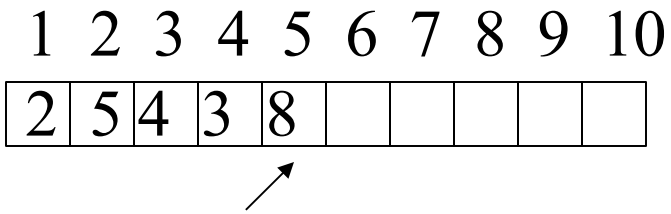
Pila (Stack)

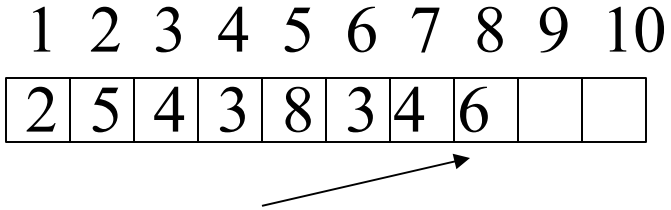


Implementazione di una Pila

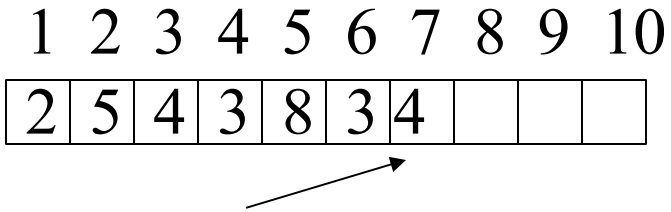
- Una Pila si può realizzare con un vettore di n elementi S
- Un attributo top[S] indica la posizione dell'elemento inserito più di recente che è anche quello da cancellare
  - oppure, con una implementazione alternativa, puo' indicare la posizione libera successiva, cioe' la posizione in cui inserire il prossimo elemento
- Quando top[S] è 0 si dice che la pila è *vuota*

Visualizzazione operazioni su Pila





Push degli elementi 3,4,6



Pop

Pseudo codice per le operazioni su Pila

```
Stack-Empty(S)
1 if top[S]=0
2       then return TRUE
3       else return FALSE

PUSH(S,x)
1 top[S]← top[S]+1
2 S[top[S]]← x

POP(S)
1 if not Stack-Empty(S)
2  then top[S]← top[S]-1
3       return S[top[S]+1]
```

Codice C++

```
#include <iostream>
#include <cassert>
using namespace std;

template<class T> class ADT_Stack{
public:
  virtual void push(T &)=0;
  virtual T pop()=0;
};
```

Codice C++

```
template<class T> class Array_Stack:
public ADT_Stack<T>{
public:
  Array_Stack(int usr_size=10):size(usr_size),top(0)
   {data=new T[size];}
  ~Array_Stack(){delete[] data;}
  void push(T &);
  T pop();
private:
  T * data;
  int size;
  int top;
};
```

Codice C++

```
template<class T>
void Array_Stack<T>::push(T & item){
  assert(top<size);
  data[top]=item;
  top++;
}

template<class T>
T Array_Stack<T>::pop(){
  top--;
  assert(top>=0);
  return data[top];
}
```

Codice C++

```
int main(){
  const int DIM=10;
  Array_Stack<int> array_stack(DIM);
  int v[]={1,2,3,4,5,6,7,8,9,10};
  ADT_Stack<int> & stack=array_stack;

  for(int i=0;i<DIM;i++)
    stack.push(v[i]);

  for(int i=0;i<DIM;i++)
    cout<<stack.pop()<<" ";
  cout<<endl;

  return 0;
}
```

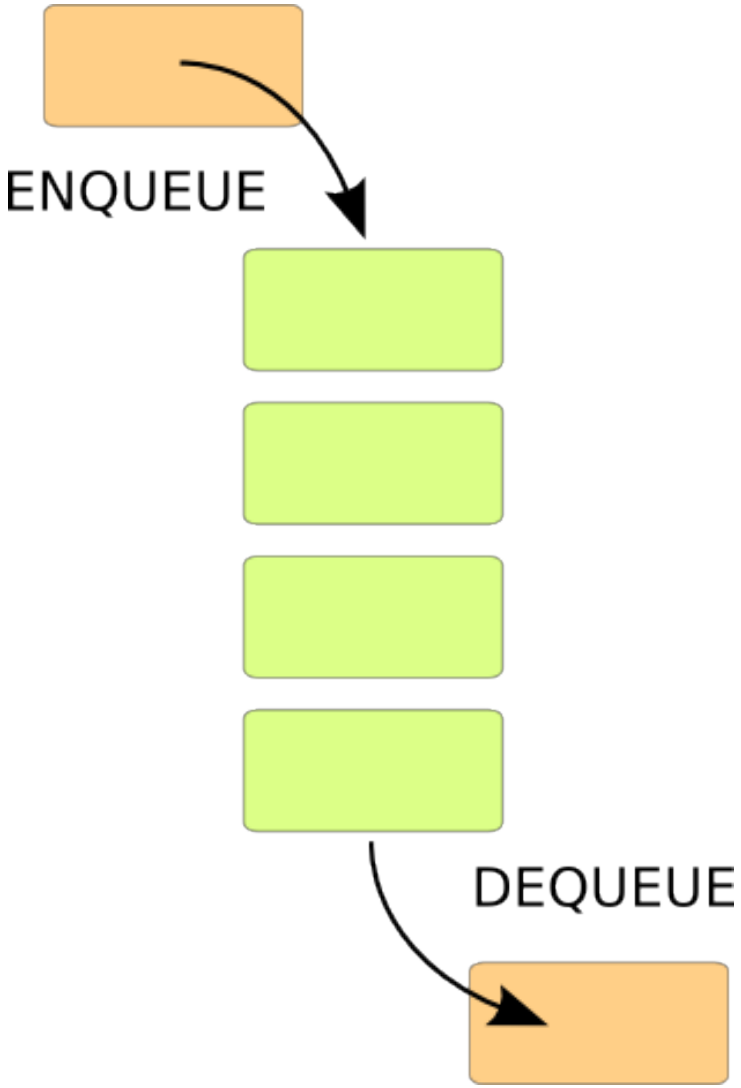
Esercizio

- Implementare la Pila con una lista invece che con un vettore
- Nota: ADT_Stack rimane immutato e nel main si cambia solo la classe da Array_Stack a List_Stack

Coda (Queue)

- La politica di cancellazione per una Coda (Queue) è di tipo FIFO (First In First Out), *primo arrivato primo servito*, cioè il primo elemento ad essere stato inserito è anche il primo elemento ad essere eliminato
- La operazione di inserzione viene chiamata Enqueue e l'operazione di eliminazione Dequeue
- Quando un nuovo elemento viene inserito prende posto in fondo alla coda

Coda (Queue)



Implementazione di una coda

- E’ possibile realizzare una coda con un vettore Q
- La coda ha attributi head[Q] e tail[Q] che rappresentano gli indici degli elementi in testa e in coda alla struttura
- L’attributo tail[Q] indica la posizione in cui sarà inserito il prossimo elemento
- Gli elementi della coda sono alle locazioni head[Q], head[Q]+1,…,tail[Q]-1
- La locazione 1 segue la locazione n (ordine circolare)
- quando head[Q]=tail[Q] la coda è vuota
- quando head[Q]=tail[Q]+1 la coda è piena

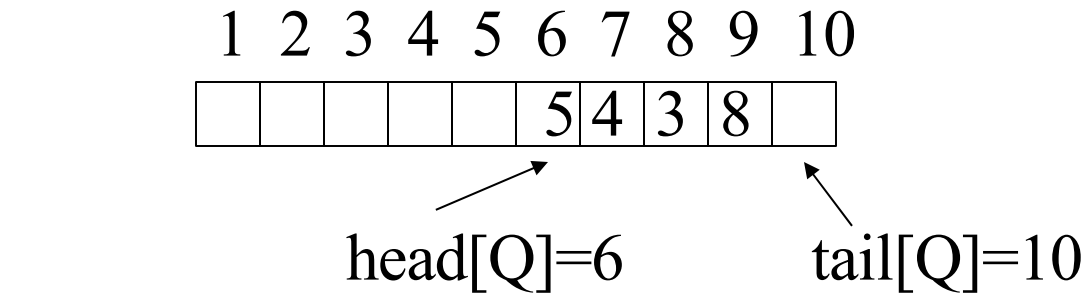
Pseudocodice coda

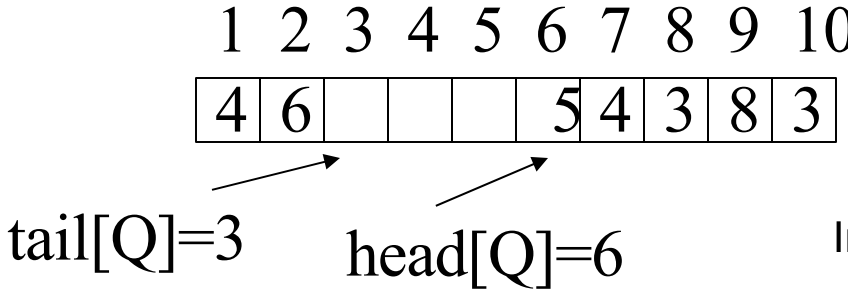
```
ENQUEUE(Q,x)
1 Q[tail[Q]]← x
2 if tail[Q]=length[Q] ▶SE SIAMO AL TERMINE DEL VETTORE
3      then tail[Q] ← 1 ▶SI INIZIA DACCAPO
4      else tail[Q] ← tail[Q]+1

DEQUEUE(x)
1 x ← Q[head[Q]]
2 if head[Q]=length[Q] ▶SE SIAMO AL TERMINE DEL VETTORE
3      then head[Q] ← 1 ▶SI INIZIA DACCAPO
4      else head[Q] ← head[Q]+1
5 return x
```

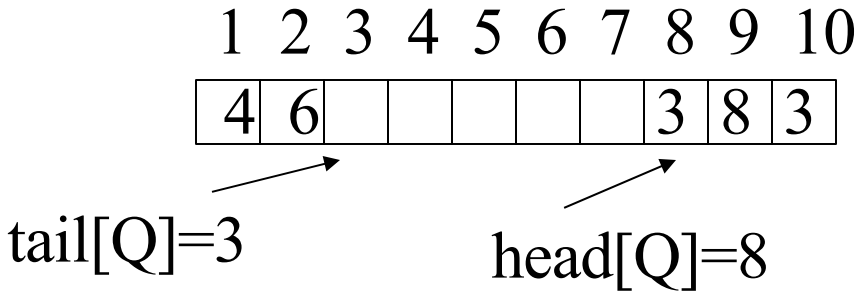
- Nota: mancano i controlli di eccedenza

Visualizzazione operazioni su coda





Inserzione degli elementi 3,4,6



Cancellazione di 2 elementi