```
//
//  queue.h
//  All
//
//  Created by Filippo Fontanelli on 02/04/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//
#include "def.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifndef _QUEUE_H
#define _QUEUE_H

/*Matrice  utilizzata come storico delle formattazine*/
int storico_formattazione[NUM_FORMATTAZIONI][2];
/*
 *Struttura che rappresenta un elemento della coda:
 * - stringa : testo
 * - link : link relativo all'eventuale url o img
 * - formattazione : Matrice di formattazione
 * - n_space : Numero di spazi presenti all'interno della stringa
 * - next : Puntatore al prossimo elemento della coda
 */
typedef struct token{
    char* stringa;
    char* link;
    int formattazione[NUM_FORMATTAZIONI][2];
    int n_space;
    struct token* next;

}token;

/*
 *Struttura che rappresenta la coda:
 * - cnt : contatore degli elementi
 * - length_string : lunghezza totale dei testi presenti nella coda
 * - tot_space : numero totale degli spazi
 * - want_opt : array relativo alle opzioni da riga di comando
 * - front : puntatore alla testa
 * - rear : puntatore alla coda
 */
typedef struct queue {
    int cnt;
    int length_string;
    int tot_space;
    int *want_opt;
    token *front;
    token *rear;

}queue;

/** Inizializza la coda con i parametri di default
 *  \param  q   coda da inizializzare
 *  \param  want_opt    array delle opzioni
 *
 *  \retval q       puntatore alla coda creata
 *  \retval ERROR   in caso di errore
 */
int init(queue *q,int *want_opt);

/** Inizializza il token con i parametri di default
 *  \param  t   token da inizializzare
```



```
 *
 *  \retval t       puntatore al token creato
 *  \retval ERROR   in caso di errore
 */
int init_token(token *t);

/** Inserisce un token dalla coda
 *  \param  t       token da inserire
 *  \param  q       coda in cui inserirlo
 *
 *  \retval 0       Token correttamente inserito
 *  \retval ERROR   in caso di errore
 */
int enqueue(token *t, queue *q);

/** Estrae un token dalla coda
 *  \param  q       coda
 *
 *  \retval t       token
 *  \retval NULL    in caso di errore
 *
 */
token* dequeue(queue *q);

/** Restituisce la testa della coda
 *  \param  q       coda
 *
 *  \retval t       token
 *  \retval NULL    in caso di errore
 *
 */
token* rear(const queue *q);

/** Restituisce la coda della coda
 *  \param  q       coda
 *
 *  \retval t       token
 *  \retval NULL    in caso di errore
 *
 */
token* front(const queue *q);

/** Verifica se la coda e' vuota
 *  \param  q       coda
 *
 *  \retval 1   coda non vuota
 *  \retval 0   coda vuota
 */
int empty(const queue *q);

/** Verifica se la coda e' piena
 *  \param  q       coda
 *
 *  \retval 1   coda non piena
 *  \retval 0   coda piena
 */
int full(const queue *q);

/** Libera la memoria occupata dal token
 *  \param  t   token
 *
 *  \retval 0   successo
 *  \retval ERROR   in caso di errore
 */
```

```
int free_token(token* t);
/** Libera la memoria occupata dalla coda
 *  \param  q       coda
 *
 *  \retval 0   successo
 *  \retval ERROR   in caso di errore
 */
int free_queue(queue *q);


#endif
```