



Corso di Programmazione e strutture dati 2022/23

Docenti di Laboratorio:

- Mattia De Rosa matderosa@unisa.it
- Michele Mastroianni mmastroianni@unisa.it

ESERCITAZIONE 5: ADT PLAYLIST

1. Esercizio 1: estendere ADT «Lista» vista in teoria
2. Esercizio 2: progettare e implementare un ADT «Playlist» usando come base l'ADT «Lista»



OVERVIEW

ESERCIZIO 1

Estendere l'ADT Lista con l'aggiunta di un nuovo operatore chiamato **sortList**:

- Progettare e implementare sortList. L'operatore deve ordinare una lista in maniera crescente

Suggerimenti:

- per realizzare l'ordinamento si utilizzi il selection sort
- Realizzare gli scambi semplicemente scambiando gli item dei nodi

ESERCIZIO 2

Specificare e realizzare un ADT «Playlist» che dovrà fornire operatori per:

1. Creare una playlist (indicata da un nome e una lista di canzoni);
 - Una canzone è caratterizzata dal titolo, dal nome del cantante (o gruppo musicale) e dalla durata della canzone in secondi.
2. Inserire una canzone in testa ad una playlist;
3. Rimuovere una canzone da una playlist dato il titolo;
4. Ordinare la playlist in base al titolo delle canzoni

Sperimentare le funzionalità dell'ADT:

1. Sviluppare un programma che istanzia e popola una nuova playlist con le seguenti caratteristiche:
 - Nome della playlist: «Rock»
 - 5 pezzi inseriti dall'utente.
2. Successivamente stampare la playlist con le canzoni ordinate

ADT: PLAYLIST

Sintattica	Semantica
Nome del tipo: Playlist Tipi usati: Song, String, boolean	Dominio: insieme di coppie <name, songs> <i>name</i> è una stringa, <i>songs</i> è una lista di Song
createPlaylist(String) → Playlist	createPlaylist(name) → pl <ul style="list-style-type: none">Post: pl = <name, nil>
addSong(Playlist, Song) → Playlist	addSong(pl, s) → pl' <ul style="list-style-type: none">Post: pl.songs = <a1, a2, ..., an> AND pl'.songs = <s, a1, ..., an>
removeSong(Playlist, String) → Playlist	removeSong(pl, e.title) → pl' <ul style="list-style-type: none">Pre: pl.songs = <a1, a2, ..., e, ..., an> n > 0Post: pl'.songs = pl.songs - <e>
sortSongs(Playlist) → Playlist	sortSongs(pl) → pl' <ul style="list-style-type: none">Pre: n > 0Post: pl'.songs = <a1, a2, ..., an> titolo(ai) < titolo(ai+1) per ogni 1 ≤ i ≤ n

ADT PLAYLIST: PROGETTAZIONE

1. Tipi preesistenti:

- Item (realizzare item-song)
- List

2. Nuovi tipi di dati:

- Song
- Playlist



```
struct playlist{  
    char *name;  
    List songs;  
};
```

```
struct list {  
    int size;  
    struct node *head;  
};
```

```
struct song{  
    char *title;  
    char *artist;  
    int duration;  
};
```

item-song.c

```
struct node {  
    Item item;  
    struct node *next;  
};
```

```
struct node {  
    Item item;  
    struct node *next;  
};
```

```
struct node {  
    Item item;  
    struct node *next;  
};
```

```
graph TD; playlist[struct playlist] -- "List songs" --> list[struct list]; list -- "struct node *head" --> node1[struct node]; node1 -- "struct node *next" --> node2[struct node]; node2 -- "struct node *next" --> node3[struct node]; song[struct song] --- item_song_c[item-song.c]; node1 --- item_song_c;
```

PLAYLIST.H

```
1  #include "song.h"
2
3  typedef struct playlist *Playlist;
4
5  Playlist createPlaylist(char* name);
6  void addSong(Playlist, Song);
7  void removeSong(Playlist, char*);
8  void sortPlaylist(Playlist);
9  void printPlaylist(Playlist);
```


SONG.H

```
1  typedef struct song *Song;
2
3  Song initSong(char* title, char* artist, int duration);
4  char* title(Song);
5  char* artist(Song);
6  int duration(Song);
```