

# Piscina C

## C 12

*Sommario: Questo documento tratta il modulo C 12 della Piscina C @ 42.*

# Indice

I	Preambolo	2
II	Istruzioni	4
III	Esercizio 00 : ft_create_elem	6
IV	Esercizio 01 : ft_list_push_front	7
V	Esercizio 02 : ft_list_size	8
VI	Esercizio 03 : ft_list_last	9
VII	Esercizio 04 : ft_list_push_back	10
VIII	Esercizio 05 : ft_list_push_strs	11
IX	Esercizio 06 : ft_list_clear	12
X	Esercizio 07 : ft_list_at	13
XI	Esercizio 08 : ft_list_reverse	14
XII	Esercizio 09 : ft_list_foreach	15
XIII	Esercizio 10 : ft_list_foreach_if	16
XIV	Esercizio 11 : ft_list_find	17
XV	Esercizio 12 : ft_list_remove_if	18
XVI	Esercizio 13 : ft_list_merge	19
XVII	Esercizio 14 : ft_list_sort	20
XVIII	Esercizio 15 : ft_list_reverse_fun	21
XIX	Esercizio 16 : ft_sorted_list_insert	22
XX	Esercizio 17 : ft_sorted_list_merge	23

# Capitolo I

## Preambolo

SPOILER ALERT  
DON'T READ THE NEXT PAGE

## You've been warned.

- In Star Wars, Dark Vador is Luke's Father.
- In The Usual Suspects, Verbal is Keyser Soze.
- In Fight Club, Tyler Durden and the narrator are the same person.
- In Sixth Sens, Bruce Willis is dead since the beginning.
- In The others, the inhabitants of the house are ghosts and vice-versa.
- In Bambi, Bambi's mother dies.
- In The Village, monsters are the villagers and the movie actually takes place in our time.
- In Harry Potter, Dumbledore dies.
- In Planet of apes, the movie takes place on earth.
- In Game of thrones, Robb Stark and Joffrey Baratheon die on their wedding day.
- In Twilight, Vampires shine under the sun.
- In Stargate SG-1, Season 1, Episode 18, O'Neill and Carter are in Antartica.
- In The Dark Knight Rises, Miranda Tate is Talia Al'Gul.
- In Super Mario Bros, The princess is in another castle.

# Capitolo II

## Istruzioni

- Fate riferimento solo a questa pagina: non fidatevi delle dicerie.
- Questo documento può subire variazioni prima della scadenza per la presentazione.
- Controllate i permessi dei vostri file e delle vostre cartelle.
- Dovete seguire le procedure di presentazione per tutti gli esercizi.
- I vostri esercizi saranno controllati e valutati dai vostri compagni di corso.
- Moulinette sarà estremamente meticolosa e severa nel valutare il vostro lavoro. Essendo il suo un processo automatico senza possibilità di ricorso, assicuratevi di essere il più precisi possibile al fine di evitare brutte sorprese.
- I vostri esercizi saranno soggetti, oltre alla valutazione tra pari, al controllo e alla valutazione da parte di un programma chiamato Moulinette.
- Moulinette non ha una mentalità aperta. Non proverà a comprendere il vostro codice se non rispetta la Norma. Moulinette utilizza un programma di nome **norminette** per controllare la validità dei vostri file. TL;DR: sarebbe scocco tentare di consegnare un esercizio che non passi il controllo di **norminette**.
- Gli esercizi sono presentati seguendo un ordine di difficoltà crescente. Ai fini della valutazione **NON** si prendono in considerazione gli esercizi se i precedenti non sono stati completati correttamente
- Usare una funzione non autorizzata viene considerato come barare. Chi bara ottiene un -42 senza possibilità di ricorso.
- Dovrete consegnare una funzione `main()` solo se l'esercizio richiede un programma.
- Moulinette compila per mezzo di `gcc` utilizzando queste flag: `-Wall -Wextra -Werror`.
- Se il vostro programma non compila, il voto sarà 0.
- NON sarà tollerato ALCUN file aggiuntivo nelle cartelle presentate oltre a quelli specificati in questo documento.


- Dubbi o domande? Chiedi a chi si trova alla tua destra, altrimenti a chi si trova alla tua sinistra
- Your reference guide is called `Google / man / the Internet / ....`
- Date un'occhiata alla sezione Piscina C del forum dell'Intranet.
- Prestate attenzione agli esempi proposti, in quanto potrebbero mostrare dettagli non esplicitamente presentati nel documento...
- Per Odin, Per Thor ! Usate la testa !!!
- Per gli esercizi presenti in questo documento dovreste usare la seguente struttura :

```
typedef struct          s_list
{
    struct s_list      *next;
    void               *data;
}                      t_list;
```

- Dovrà essere inclusa nel file `ft_list.h` da consegnare per ogni esercizio.
- Useremo il nostro `ft_create_elem` dall'esercizio 01 in poi, agite quindi di conseguenza (potrebbe essere utile averne il prototipo nel file `ft_list.h...`).

# Capitolo III

## Esercizio 00 : ft\_create\_elem


	Esercizio 00
	ft_create_elem
Cartella per la consegna : <i>ex00/</i>	
File da consegnare : <b>ft_create_elem.c</b> , <b>ft_list.h</b>	
Funzioni permesse : <b>malloc</b>	

- Creare la funzione **ft\_create\_elem** che crea un nuovo elemento di tipo **t\_list**.
- Deve assegnare come valore a **data** quello dell'argomento dato e a **next** quello di **NULL**.
- Il prototipo è il seguente :

```
t_list      *ft_create_elem(void *data);
```

# Capitolo IV

## Esercizio 01 : ft\_list\_push\_front

	Esercizio 01
	ft_list_push_front
	Cartella per la consegna : <i>ex01/</i>
	File da consegnare : <code>ft_list_push_front.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>ft_create_elem</code>


- Creare la funzione `ft_list_push_front` che aggiunge un nuovo elemento di tipo `t_list` all'inizio della lista.
- Deve assegnare come valore a `data` quello dell'argomento dato.
- Se ve ne è la necessità deve aggiornare il puntatore all'inizio della lista.
- Il prototipo è il seguente :

```
void      ft_list_push_front(t_list **begin_list, void *data);
```



# Capitolo V

## Esercizio 02 : ft\_list\_size


	Esercizio 02
	ft_list_size
	Cartella per la consegna : <i>ex02/</i>
	File da consegnare : <code>ft_list_size.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_size` che restituisce il numero di elementi presenti nella lista.
- Il prototipo è il seguente :

```
int ft_list_size(t_list *begin_list);
```

# Capitolo VI

## Esercizio 03 : ft\_list\_last


	Esercizio 03
	ft_list_last
	Cartella per la consegna : <i>ex03/</i>
	File da consegnare : <code>ft_list_last.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_last` che restituisce l'ultimo elemento presente nella lista.
- Il prototipo è il seguente :

```
t_list *ft_list_last(t_list *begin_list);
```

# Capitolo VII

## Esercizio 04 : ft\_list\_push\_back


	Esercizio 04
	ft_list_push_back
	Cartella per la consegna : <i>ex04/</i>
	File da consegnare : <code>ft_list_push_back.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>ft_create_elem</code>

- Creare la funzione `ft_list_push_back` che aggiunge un nuovo elemento di tipo `t_list` alla fine della lista.
- Deve assegnare come valore a `data` quello dell'argomento dato.
- Se ve ne è la necessità deve aggiornare il puntatore all'inizio della lista.
- Il prototipo è il seguente :

```
void      ft_list_push_back(t_list **begin_list, void *data);
```

# Capitolo VIII

## Esercizio 05 : ft\_list\_push\_strs


	Esercizio 05
	ft_list_push_strs
	Cartella per la consegna : <i>ex05/</i>
	File da consegnare : <code>ft_list_push_strs.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>ft_create_elem</code>

- Creare la funzione `ft_list_push_strs` che crea una nuova lista contenente tutte le stringhe puntate da `strs`.
- `size` è la dimensione di `strs`
- Il primo elemento deve essere posizionato alla fine della lista.
- Deve restituire l'indirizzo del primo link della lista.
- Il prototipo è il seguente :

```
t_list *ft_list_push_strs(int size, char **strs);
```

# Capitolo IX

## Esercizio 06 : ft\_list\_clear

	Esercizio 06
	ft_list_clear
	Cartella per la consegna : <i>ex06/</i>
	File da consegnare : <code>ft_list_clear.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>free</code>

- Creare la funzione `ft_list_clear` che elimina e libera tutti i link presenti nella lista.
- Userete `free_fct` per liberare ogni elemento `data`
- Il prototipo è il seguente :

```
void ft_list_clear(t_list *begin_list, void (*free_fct)(void *));
```

# Capitolo X

## Esercizio 07 : ft\_list\_at

	Esercizio 07
	ft_list_at
	Cartella per la consegna : <i>ex07/</i>
	File da consegnare : <code>ft_list_at.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_at` che restituisca l'elemento `n` della lista, considerando che per `nbr = 0`, restituirà il primo elemento della lista.
- In caso di errore, restituirà un puntatore nullo.
- Il prototipo è il seguente :

```
t_list *ft_list_at(t_list *begin_list, unsigned int nbr);
```

# Capitolo XI

## Esercizio 08 : ft\_list\_reverse


	Esercizio 08
	ft_list_reverse
	Cartella per la consegna : <i>ex08/</i>
	File da consegnare : <b>ft_list_reverse.c</b>
	Funzioni permesse : Nessuna

- Creare la funzione **ft\_list\_reverse** che inverte l'ordine degli elementi di una lista. Il valore di ogni elemento deve rimanere invariato.
- Tenete a mente che utilizzeremo il nostro **ft\_list.h**
- Il prototipo è il seguente :

```
void ft_list_reverse(t_list **begin_list);
```

# Capitolo XII

## Esercizio 09 : ft\_list\_foreach

	Esercizio 09
	ft_list_foreach
	Cartella per la consegna : <i>ex09/</i>
	File da consegnare : <b>ft_list_foreach.c</b> , <b>ft_list.h</b>
	Funzioni permesse : Nessuna

- Creare la funzione **ft\_list\_foreach** che applica, ad ogni elemento della lista, la funzione data come argomento.
- **f** deve essere applicata seguendo l'ordine della lista.
- Il prototipo è il seguente :

```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```


- la funzione puntata da **f** sarà utilizzata come segue :

```
(*f)(list_ptr->data);
```



# Capitolo XIII

## Esercizio 10 : ft\_list\_foreach\_if

	Esercizio 10
	ft_list_foreach_if
	Cartella per la consegna : <i>ex10/</i>
	File da consegnare : <code>ft_list_foreach_if.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_foreach_if` che applica, ad alcuni elementi della lista, la funzione `data` come argomento.
- La funzione viene applicata solo a quegli elementi per cui `cmp` restituisce 0.
- `f` deve essere applicata seguendo l'ordine della lista.
- Il prototipo è il seguente :

```
void      ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void  
*data_ref, int (*cmp)())
```

- Le funzioni puntate da `f` e da `cmp` saranno utilizzate come segue :

```
(*f)(list_ptr->data);  
(*cmp)(list_ptr->data, data_ref);
```



Per esempio `cmp` potrebbe essere `ft_strcmp...`

# Capitolo XIV

## Esercizio 11 : ft\_list\_find

	Esercizio 11
	ft_list_find
	Cartella per la consegna : <i>ex11/</i>
	File da consegnare : <code>ft_list_find.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_find` che restituisce l'indirizzo del primo elemento della lista per cui `cmp` restituisce 0.
- Il prototipo è il seguente :


```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

- La funzione puntata da `cmp` sarà utilizzata come segue :

```
(*cmp)(list_ptr->data, data_ref);
```

# Capitolo XV

## Esercizio 12 : ft\_list\_remove\_if

	Esercizio 12
	ft_list_remove_if
	Cartella per la consegna : <i>ex12/</i>
	File da consegnare : <code>ft_list_remove_if.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>free</code>

- Creare la funzione `ft_list_remove_if` che elimina ogni elemento della lista per cui `cmp` restituisce 0.
- Ogni elemento eliminato dalla lista deve essere liberato usando `free_fct`
- Il prototipo è il seguente :


```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *))
```

- Le funzioni puntate da `cmp` e da `free_fct` saranno utilizzate come segue :

```
(*cmp)(list_ptr->data, data_ref);  
(*free_fct)(list_ptr->data);
```

# Capitolo XVI

## Esercizio 13 : ft\_list\_merge


	Esercizio 13
	ft_list_merge
	Cartella per la consegna : <i>ex13/</i>
	File da consegnare : <code>ft_list_merge.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_merge` che posiziona gli elementi di una lista `begin2` alla fine di un'altra lista `begin1`.
- La creazione di elementi non è consentita.
- Il prototipo è il seguente :

```
void ft_list_merge(t_list **begin_list1, t_list *begin_list2);
```

# Capitolo XVII

## Esercizio 14 : ft\_list\_sort

	Esercizio 14
	ft_list_sort
	Cartella per la consegna : <i>ex14/</i>
	File da consegnare : <code>ft_list_sort.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_sort` che ordina gli elementi di una lista in ordine crescente utilizzando una funzione per comparare i dati di due elementi per volta.
- Il prototipo è il seguente :

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

- La funzione puntata da `cmp` sarà utilizzata come segue:


```
(*cmp)(list_ptr->data, list_other_ptr->data);
```



`cmp` potrebbe essere `ft_strcmp`.

# Capitolo XVIII

## Esercizio 15 : ft\_list\_reverse\_fun


	Esercizio 15
	ft_list_reverse_fun
	Cartella per la consegna : <i>ex15/</i>
	File da consegnare : <code>ft_list_reverse_fun.c</code> , <code>ft_list.h</code>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_list_reverse_fun` che inverte l'ordine degli elementi della lista.
- Il prototipo è il seguente :

```
void ft_list_reverse_fun(t_list *begin_list);
```

# Capitolo XIX

## Esercizio 16 : ft\_sorted\_list\_insert

	Esercizio 16
	ft_sorted_list_insert
	Cartella per la consegna : <i>ex16/</i>
	File da consegnare : <code>ft_sorted_list_insert.c</code> , <code>ft_list.h</code>
	Funzioni permesse : <code>ft_create_elem</code>

- Creare la funzione `ft_sorted_list_insert` che crea un nuovo elemento e lo inserisce in una lista ordinata per ordine crescente mantenendone l'ordine.
- Il prototipo è il seguente :


```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

- La funzione `cmp` sarà utilizzata come segue :

```
(*cmp)(list_ptr->data, list_other_ptr->data);
```

# Capitolo XX

## Esercizio 17 : ft\_sorted\_list\_merge

	Esercizio 17
	ft_sorted_list_merge
	Cartella per la consegna : <i>ex17/</i>
	File da consegnare : <i>ft_sorted_list_merge.c, ft_list.h</i>
	Funzioni permesse : Nessuna

- Creare la funzione `ft_sorted_list_merge` che inserisce gli elementi di una lista ordinata `begin2` in un'altra lista ordinata per ordine crescente `begin1`, mantenendo l'ordine di quest'ultima.
- Il prototipo è il seguente :

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

- La funzione `cmp` sarà utilizzata come segue :l

```
(*cmp)(list_ptr->data, list_other_ptr->data);
```