



# Piscina C

## C 02

*Sommario: Questo documento tratta il modulo C 01 della Piscina C @ 42.*

# Indice

I	Istruzioni	2
II	Foreword	4
III	Esercizio 00 : ft_strcpy	5
IV	Esercizio 01 : ft_strncpy	6
V	Esercizio 02 : ft_str_is_alpha	7
VI	Esercizio 03 : ft_str_is_numeric	8
VII	Esercizio 04 : ft_str_is_lowercase	9
VIII	Esercizio 05 : ft_str_is_uppercase	10
IX	Esercizio 06 : ft_str_is_printable	11
X	Esercizio 07 : ft_strupcase	12
XI	Esercizio 08 : ft_strlowcase	13
XII	Esercizio 09 : ft_strcapitalize	14
XIII	Esercizio 10 : ft_strlcpy	15
XIV	Exercise 11 : ft_putstr_non_printable	16
XV	Exercise 12 : ft_print_memory	17
XVI	Consegna e valutazione tra pari	19

# Capitolo I

## 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 pass 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 !!!



Norminette va utilizzata con la flag `-R CheckForbiddenSourceHeader`.  
Moulinette farà la stessa cosa.

# Capitolo II

## Foreword

Here is a discuss extract from the Silicon Valley serie:

- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- Oh, God, help us! Okay, uh you know what? I just don't think this is going to work. I'm so sorry. Uh, I mean like, what, we're going to bring kids into this world with that over their heads? That's not really fair to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SLAMS) - (BANGING)

. . .


(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.

Hopefully, you are not forced to use emacs and your space bar to complete the following exercices.

# Capitolo III

## Esercizio 00 : ft\_strcpy


	Esercizio 00
ft_strcpy	
Cartella per la consegna : <i>ex00/</i>	
File da consegnare : <b>ft_strcpy.c</b>	
Funzioni permesse : Nessuna	

- Riproduci il comportamento della funzione `strcpy` (`man strcpy`).
- Il prototipo è il seguente :

```
char      *ft_strcpy(char *dest, char *src);
```

# Capitolo IV

## Esercizio 01 : ft\_strncpy


	Esercizio 01
ft_strncpy	
Cartella per la consegna : <i>ex01/</i>	
File da consegnare : <b>ft_strncpy.c</b>	
Funzioni permesse : Nessuna	

- Riproduci il comportamento della funzione **strncpy** (man strncpy).
- Il prototipo è il seguente :

```
char      *ft_strncpy(char *dest, char *src, unsigned int n);
```

# Capitolo V

## Esercizio 02 : ft\_str\_is\_alpha

	Esercizio 02
ft_str_is_alpha	
Cartella per la consegna : <i>ex02/</i>	
File da consegnare : <b>ft_str_is_alpha.c</b>	
Funzioni permesse : <b>Nessuna</b>	

- Creare una funzione che restituisca 1 se la stringa passata come parametro contiene solo caratteri alfabetici o 0 se contiene almeno un carattere di un altro tipo.
- Il prototipo è il seguente :


```
int ft_str_is_alpha(char *str);
```

- Se **str** è vuota deve restituire 1.



# Capitolo VI

## Esercizio 03 : ft\_str\_is\_numeric

	Esercizio 03
	ft_str_is_numeric
	Cartella per la consegna : <i>ex03/</i>
	File da consegnare : <b>ft_str_is_numeric.c</b>
	Funzioni permesse : <b>Nessuna</b>


- Creare una funzione che restituisca 1 se la stringa passatela come parametro contiene solo caratteri numerici o 0 se contiene almeno un carattere di un altro tipo.
- Il prototipo è il seguente :

```
int      ft_str_is_numeric(char *str);
```

- Se **str** è vuota deve restituire 1.

# Capitolo VII

## Esercizio 04 : ft\_str\_is\_lowercase

	Esercizio 04
ft_str_is_lowercase	
Cartella per la consegna : <i>ex04/</i>	
File da consegnare : <b>ft_str_is_lowercase.c</b>	
Funzioni permesse : <b>Nessuna</b>	


- Creare una funzione che restituisca 1 se la stringa passatela come parametro contiene solo caratteri alfabetici minuscoli o 0 se contiene almeno un carattere di un altro tipo.
- Il prototipo è il seguente :

```
int      ft_str_is_lowercase(char *str);
```

- Se **str** è vuota deve restituire 1.

# Capitolo VIII

## Esercizio 05 : ft\_str\_is\_uppercase

	Esercizio 05
ft_str_is_uppercase	
Cartella per la consegna : <i>ex05/</i>	
File da consegnare : <b>ft_str_is_uppercase.c</b>	
Funzioni permesse : <b>Nessuna</b>	


- Creare una funzione che restituisca 1 se la stringa passatela come parametro contiene solo caratteri alfabetici minuscoli o 0 se contiene almeno un carattere di un altro tipo.
- Il prototipo è il seguente :

```
int ft_str_is_uppercase(char *str);
```

- Se **str** è vuota deve restituire 1.

# Capitolo IX

## Esercizio 06 : ft\_str\_is\_printable

	Esercizio 06
ft_str_is_printable	
Cartella per la consegna : <i>ex06/</i>	
File da consegnare : <b>ft_str_is_printable.c</b>	
Funzioni permesse : Nessuna	


- Creare una funzione che restituisca 1 se la stringa passatela come parametro contiene solo caratteri stampabili o 0 se contiene almeno un carattere di un altro tipo.
- Il prototipo è il seguente :

```
int ft_str_is_printable(char *str);
```

- Se **str** è vuota deve restituire 1.

# Capitolo X

## Esercizio 07 : ft\_strupcase

	Esercizio 07
	ft_strupcase
	Cartella per la consegna : <i>ex07/</i>
	File da consegnare : <b>ft_strupcase.c</b>
	Funzioni permesse : Nessuna


- Creare una funzione che converte in maiuscolo ogni lettera.
- Il prototipo è il seguente :

```
char *ft_strupcase(char *str);
```

- Deve restituire **str**.

# Capitolo XI

## Esercizio 08 : ft\_strlowcase

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

- Creare una funzione che converte in minuscolo ogni lettera.
- Il prototipo è il seguente :

```
char *ft_strlowcase(char *str);
```

- Deve restituire **str**.

# Capitolo XII

## Esercizio 09 : ft\_strcapitalize

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

- Creare una funzione che converta in maiuscolo la prima lettera di ogni parola e in minuscolo le restanti.
- Una parola è una stringa di caratteri alfanumerici.
- Il prototipo è il seguente :

```
char *ft_strcapitalize(char *str);
```

- Deve restituire **str**.
- Ad esempio:


```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

- Diventa:

```
Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un
```

# Capitolo XIII

## Esercizio 10 : ft\_strlcpy

	Esercizio 10
	ft_strlcpy
	Cartella per la consegna : <i>ex10/</i>
	File da consegnare : <b>ft_strlcpy.c</b>
	Funzioni permesse : Nessuna


- Riproduci il comportamento della funzione `strlcpy` (`man strlcpy`).
- Il prototipo è il seguente :

```
unsigned int ft_strlcpy(char *dest, char *src, unsigned int size);
```



# Capitolo XIV

## Exercise 11 : ft\_putstr\_non\_printable

	Esercizio 11
ft_putstr_with_non_printable	
Cartella per la consegna : <i>ex11/</i>	
File da consegnare : <b>ft_putstr_non_printable.c</b>	
Funzioni permesse : <b>write</b>	

- Creare una funzione che stampi a video una stringa di caratteri. Se la stringa contiene caratteri non stampabili, dovranno essere stampati in forma esadecimale(lowercase) preceduti da un backslash(barra rovesciata).
- Ad esempio per :

```
Coucou\ntu vas bien ?
```

- La funzione deve stampare a video :


```
Coucou\0atu vas bien ?
```

- Il prototipo è il seguente :

```
void      ft_putstr_non_printable(char *str);
```

# Capitolo XV

## Exercise 12 : ft\_print\_memory

	Esercizio 12
ft_print_memory	
Cartella per la consegna : <i>ex12/</i>	
File da consegnare : <b>ft_print_memory.c</b>	
Funzioni permesse : <b>write</b>	

- Creare una funzione che stampi a video l'area di memoria.
- Il tutto deve essere separato in tre diverse "colonne" separate da uno spazio :
  - L'indirizzo in esadecimale del primo carattere della prima riga seguito da ':'.
    - Il contenuto in esadecimale presentato con uno spazio ogni 2 caratteri e riempito (padded) con spazi dove necessario (vedere esempio).
    - Il contenuto in caratteri stampabili.
- Se un carattere è un non-stampabile sarà sostituito da un punto.
- Ogni riga dovrà gestire 16 caratteri.
- Se **size** è 0 non verrà stampato niente.

- Esempio:

```
$> ./ft_print_memory
000000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010a161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.
000000010a161f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..
000000010a161f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol. .
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
0000000107ff9f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.$
0000000107ff9f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..$
0000000107ff9f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol. .$.
$>
```

- Il prototipo è il seguente :

```
void      *ft_print_memory(void *addr, unsigned int size);
```

- Deve restituire addr.

# Capitolo XVI

## Consegna e valutazione tra pari

Consegna gli esercizi nella tuo repository `Git` come al solito. Durante la difesa verrà considerato unicamente ciò che si trova all'interno della repository. Assicurati di controllare che i nomi dei tuoi file siano corretti.



Devi consegnare solo i file richiesti da questo documento.