



C Piscine

C 02

*Summary: This document is the subject for the C 02 module of the C Piscine @ 42.*

*Version: 5.5*

# Contents

|      |                                       |    |
|------|---------------------------------------|----|
| I    | Instructions                          | 2  |
| II   | Foreword                              | 4  |
| III  | Exercise 00 : ft_strcpy               | 6  |
| IV   | Exercise 01 : ft_strncpy              | 7  |
| V    | Exercise 02 : ft_str_is_alpha         | 8  |
| VI   | Exercise 03 : ft_str_is_numeric       | 9  |
| VII  | Exercise 04 : ft_str_is_lowercase     | 10 |
| VIII | Exercise 05 : ft_str_is_uppercase     | 11 |
| IX   | Exercise 06 : ft_str_is_printable     | 12 |
| X    | Exercise 07 : ft_strupcase            | 13 |
| XI   | Exercise 08 : ft_strlowcase           | 14 |
| XII  | Exercise 09 : ft_strcapitalize        | 15 |
| XIII | Exercise 10 : ft_strlcpy              | 16 |
| XIV  | Exercise 11 : ft_putstr_non_printable | 17 |
| XV   | Exercise 12 : ft_print_memory         | 18 |
| XVI  | Submission and peer-evaluation        | 20 |

# Chapter I

## Instructions

- Only this page serves as your reference, do not trust rumors.
- Watch out! This document may change before submission.
- Ensure you have the appropriate permissions on your files and directories.
- You must follow the **submission procedures** for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be evaluated by a program called **Moulinette**.
- **Moulinette** is meticulous and strict in its assessment. It is fully automated, and there is no way to negotiate with it. To avoid unpleasant surprises, be as thorough as possible.
- **Moulinette** is not open-minded. If your code does not adhere to the Norm, it won't attempt to understand it. **Moulinette** relies on a program called **norminette** to check if your files comply with the Norm. TL;DR: Submitting work that doesn't pass **norminette**'s check makes no sense.
- These exercises are arranged in order of difficulty, from easiest to hardest. We **will not** consider a successfully completed harder exercise if an easier one is not fully functional.
- Using a forbidden function is considered cheating. Cheaters receive a grade of **-42**, which is non-negotiable.
- You only need to submit a **main()** function if we specifically ask for a **program**
- **Moulinette** compiles with the following flags: **-Wall -Wextra -Werror**, using **cc**.
- If your program does not compile, you will receive a grade of **0**.
- You **cannot** leave **any** additional file in your directory beyond those specified in the assignment.
- Have a question? Ask the peer on your right. If not, try the peer on your left.

- Your reference guide is called **Google / man / the Internet / ...**
- Check the "C Piscine" section of the forum on the intranet or the Piscine on Slack.
- Carefully examine the examples. They may contain crucial details that are not explicitly stated in the assignment...
- By Odin, by Thor! Use your brain!!!



Norminette must be run with the `-R CheckForbiddenSourceHeader` flag, which will also be used by Moulinette.

# Chapter II

## Foreword

Here is an excerpt from a discussion in 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 required to use emacs and your space bar to complete the following exercises.


## Today's threshold

The validation threshold for this project is 50%.

It is up to you to determine which exercises allow you to reach this threshold and whether you want to complete additional exercises.

# Chapter III

## Exercise 00 : ft\_strcpy


|   |             |
|---|-------------|
|  | Exercise 00 |
| ft_strcpy   |             |
| Turn-in directory : <i>ex00/</i>  |             |
| Files to turn in : <b>ft_strcpy.c</b>   |             |
| Allowed functions : None  |             |

- Reproduce the behavior of the function `strcpy` (man `strcpy`).
- Here is how it should be prototyped:

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

# Chapter IV

## Exercise 01 : ft\_strncpy

|   |             |
|---|-------------|
|  | Exercise 01 |
| ft_strncpy  |             |
| Turn-in directory : <i>ex01/</i>  |             |
| Files to turn in : <b>ft_strncpy.c</b>  |             |
| Allowed functions : None  |             |


- Reproduce the behavior of the function `strncpy` (man `strncpy`).
- Here is how it should be prototyped:

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



# Chapter V

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

|   |             |
|---|-------------|
|  | Exercise 02 |
| ft_str_is_alpha   |             |
| Turn-in directory : <i>ex02/</i>  |             |
| Files to turn in : <b>ft_str_is_alpha.c</b>                                       |             |
| Allowed functions : None  |             |


- Create a function that returns 1 if the given string contains only alphabetical characters and 0 if it contains any other character.
- Here is how it should be prototyped:

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

- It should return 1 if **str** is empty.

# Chapter VI

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

|   |             |
|---|-------------|
|  | Exercise 03 |
| ft_str_is_numeric   |             |
| Turn-in directory : <i>ex03/</i>  |             |
| Files to turn in : <b>ft_str_is_numeric.c</b>                                     |             |
| Allowed functions : <b>None</b>   |             |


- Create a function that returns 1 if the given string contains only digits and 0 if it contains any other character.
- Here is how it should be prototyped:

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

- It should return 1 if **str** is empty.

# Chapter VII

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

|   |             |
|---|-------------|
|  | Exercise 04 |
| ft_str_is_lowercase   |             |
| Turn-in directory : <i>ex04/</i>  |             |
| Files to turn in : <b>ft_str_is_lowercase.c</b>                                   |             |
| Allowed functions : <b>None</b>   |             |


- Create a function that returns 1 if the given string contains only lowercase alphabetical characters and 0 if it contains any other character.
- Here is how it should be prototyped:

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

- It should return 1 if **str** is empty.

# Chapter VIII

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

|   |             |
|---|-------------|
|  | Exercise 05 |
| ft_str_is_uppercase   |             |
| Turn-in directory : <i>ex05/</i>  |             |
| Files to turn in : <b>ft_str_is_uppercase.c</b>                                   |             |
| Allowed functions : None  |             |


- Create a function that returns 1 if the given string contains only uppercase alphabetical characters and 0 if it contains any other character.
- Here is how it should be prototyped:

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

- It should return 1 if **str** is empty.

# Chapter IX

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

|   |             |
|---|-------------|
|  | Exercise 06 |
| ft_str_is_printable   |             |
| Turn-in directory : <i>ex06/</i>  |             |
| Files to turn in : <b>ft_str_is_printable.c</b>                                   |             |
| Allowed functions : None  |             |


- Create a function that returns 1 if the given string contains only printable characters and 0 if it contains any other character.
- Here is how it should be prototyped:

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

- It should return 1 if **str** is empty.

# Chapter X

## Exercise 07 : ft\_strupcase

|   |             |
|---|-------------|
|  | Exercise 07 |
| ft_strupcase  |             |
| Turn-in directory : <i>ex07/</i>  |             |
| Files to turn in : <b>ft_strupcase.c</b>  |             |
| Allowed functions : None  |             |


- Create a function that converts every letter to uppercase.
- Here is how it should be prototyped:

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

- It should return **str**.

# Chapter XI

## Exercise 08 : ft\_strlowercase

|   |             |
|---|-------------|
|  | Exercise 08 |
| ft_strlowercase   |             |
| Turn-in directory : <i>ex08/</i>  |             |
| Files to turn in : <b>ft_strlowercase.c</b>                                       |             |
| Allowed functions : None  |             |


- Create a function that converts every letter to lowercase.
- Here is how it should be prototyped:

```
char *ft_strlowercase(char *str);
```

- It should return **str**.

# Chapter XII

## Exercise 09 : ft\_strcapitalize

|   |             |
|---|-------------|
|  | Exercise 09 |
| ft_strcapitalize  |             |
| Turn-in directory : <i>ex09/</i>  |             |
| Files to turn in : <b>ft_strcapitalize.c</b>                                      |             |
| Allowed functions : None  |             |

- Create a function that capitalizes the first letter of each word and converts all other letters to lowercase.
- A word is a sequence of alphanumeric characters.
- Here is how it should be prototyped:

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

- It should return `str`.
- For example:

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


- Becomes:

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



# Chapter XIII

## Exercise 10 : ft\_strlcpy


|   |             |
|---|-------------|
|  | Exercise 10 |
| ft_strlcpy  |             |
| Turn-in directory : <i>ex10/</i>  |             |
| Files to turn in : <b>ft_strlcpy.c</b>  |             |
| Allowed functions : None  |             |

- Reproduce the behavior of the function `strlcpy` (man `strlcpy`).
- Here is how it should be prototyped:

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

# Chapter XIV

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

|   |             |
|---|-------------|
|  | Exercise 11 |
| ft_putstr_non_printable   |             |
| Turn-in directory : <i>ex11/</i>  |             |
| Files to turn in : <b>ft_putstr_non_printable.c</b>                               |             |
| Allowed functions : <b>write</b>  |             |

- Create a function that displays a string of characters on screen. If this string contains non-printable characters, they must be displayed as lowercase hexadecimal values, preceded by a backslash.
- For example:

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

- The function should display:


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

- Here is how it should be prototyped:

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

# Chapter XV

## Exercise 12 : ft\_print\_memory

|   |             |
|---|-------------|
|  | Exercise 12 |
| ft_print_memory   |             |
| Turn-in directory : <i>ex12/</i>  |             |
| Files to turn in : <b>ft_print_memory.c</b>                                       |             |
| Allowed functions : <b>write</b>  |             |

- Create a function that displays a memory area on screen.
- The display of this memory area should be divided into three "columns", separated by a space:
  - The hexadecimal address of the first character in the line, followed by a ':'.
    - The content in hexadecimal, with a space every two characters, and padded with spaces if necessary (see the example below).
    - The content in printable characters.
- If a character is non-printable, it should be replaced by a dot
- Each line should display sixteen characters.
- If **size** is equal to 0, nothing should be displayed.

- Example:

```
$> ./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. .$.
$>
```

- Here is how it should be prototyped:

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

- It should return addr.

# Chapter XVI

## Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double-check the names of your files to ensure they are correct.



You must submit only the files required by the subject of this project.