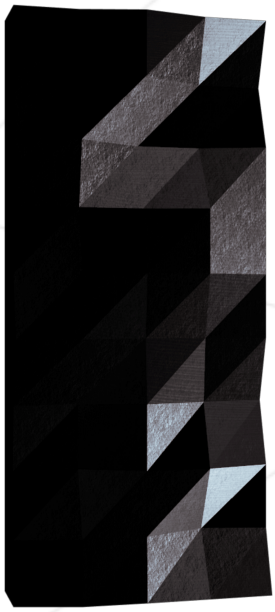

C Piscine

C 02

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

Version: 6

Contents

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

AI Instructions

Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

Main message

- Build strong foundations without shortcuts.
- Really develop tech & power skills.
- Experience real peer-learning, start learning how to learn and solve new problems.
- The learning journey is more important than the result.
- Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your projects. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter III

Foreword

Here is an excerpt from a discussion in the *Silicon Valley* series:

```
- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emacs.
- 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 IV

Exercise 00 : ft_strcpy

	Exercise 00
ft_strcpy	
Turn-in directory: $ex00/$	
Files to turn in: `ft_strcpy.c`	
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 V

Exercise 01 : ft_strncpy

	Exercise 01
ft_strncpy	
Turn-in directory: *ex*01/	
Files to turn in: `ft_strncpy.c`	
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 VI

Exercise 02 : ft_str_is_alpha

	Exercise 02
ft_str_is_alpha	
Turn-in directory: *ex*02/	
Files to turn in: `ft_str_is_alpha.c`	
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 VII

Exercise 03 : ft_str_is_numeric

	Exercise 03
ft_str_is_numeric	
Turn-in directory: *ex*03/	
Files to turn in: `ft_str_is_numeric.c`	
Allowed functions: `None`	

- 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 VIII

Exercise 04 : ft_str_is_lowercase

	Exercise 04
ft_str_is_lowercase	
Turn-in directory: *ex*04/	
Files to turn in: `ft_str_is_lowercase.c`	
Allowed functions: `None`	

- 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 IX

Exercise 05 : ft_str_is_uppercase

	Exercise 05
ft_str_is_uppercase	
Turn-in directory: $ex05/$	
Files to turn in: `ft_str_is_uppercase.c`	
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 X

Exercise 06 : ft_str_is_printable

	Exercise 06
ft_str_is_printable	
Turn-in directory: $ex06/$	
Files to turn in: `ft_str_is_printable.c`	
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 XI

Exercise 07 : ft_strupcase

	Exercise 07
ft_strupcase	
Turn-in directory: $ex07/$	
Files to turn in: `ft_strupcase.c`	
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 XII

Exercise 08 : ft_strlowcase

	Exercise 08
ft_strlowcase	
Turn-in directory: *ex*08/	
Files to turn in: `ft_strlowcase.c`	
Allowed functions: `None`	

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

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

- It should return `str`.

Chapter XIII

Exercise 09 : ft_strcapitalize

	Exercise 09
ft_strcapitalize	
Turn-in directory: *ex*09/	
Files to turn in: `ft_strcapitalize.c`	
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:

```
hi, how are you? 42words forty-two; fifty+and+one
```

- Becomes:

```
Hi, How Are You? 42words Forty-Two; Fifty+And+One
```

Chapter XIV

Exercise 10 : ft_strlcpy

	Exercise 10
ft_strlcpy	
Turn-in directory: $ex10/$	
Files to turn in: `ft_strlcpy.c`	
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 XV

Exercise 11 : ft_putstr_non_printable

	Exercise 11
ft_putstr_non_printable	
Turn-in directory: $ex11/$	
Files to turn in: `ft_putstr_non_printable.c`	
Allowed functions: `write`	

- 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:

```
Hello\nHow are you?
```

- The function should display:

```
Hello\0aHow are you?
```

- Here is how it should be prototyped:

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

Chapter XVI

Exercise 12 : ft_print_memory

	Exercise 12
ft_print_memory	
Turn-in directory: $ex12/$	
Files to turn in: `ft_print_memory.c`	
Allowed functions: `write`	

- 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 XVII

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.`