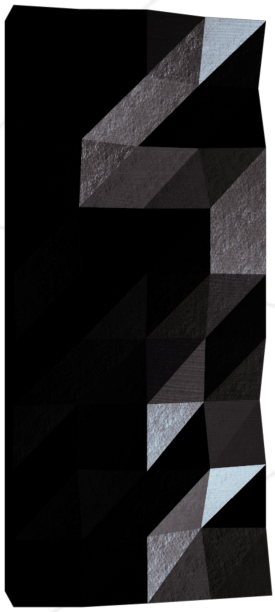

C Piscine

C 00

Summary: This document serves as the subject for the C 00 module of the C Piscine at 42.

Version: 8

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!!!

Don't forget to include the *standard 42 header* in each of your `.c` and `.h` files. Norminette checks for its presence anyway!

Norminette must be run with the *-R CheckForbiddenSourceHeader* flag. Moulinette will use it too.

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

```
Cod liver oil is a nutritional supplement derived from the liver of codfish (Gadidae

Like most fish oils, it contains high levels of omega-3 fatty acids, including
eicosapentaenoic acid (EPA) and docosahexaenoic acid (DHA). Cod liver oil is
also rich in vitamins A and D.

Historically, it has been consumed for its vitamin A and vitamin D content.

It was once commonly given to children, as vitamin D has been shown to prevent
rickets and other symptoms of vitamin D deficiency.
```

In contrast to cod liver oil, C is good, make sure to eat some!

Chapter IV

Exercise 00: ft_putchar

	Exercise 00
ft_putchar	
Turn-in directory: *ex*00/	
Files to turn in: `ft_putchar.c`	
Allowed functions: `write`	

- Write a function that displays the character passed as a parameter.
- The function should be prototyped as follows:

```
void ft_putchar(char c);
```

- To display the character, you must use the **write** function as follows:

```
write(1, &c, 1);
```

The first retry delay is short, so don't hesitate to trigger an intermediate evaluation to track your progress.

Chapter V

Exercise 01: ft_print_alphabet

	Exercise 01
ft_print_alphabet	
Turn-in directory: *ex*01/	
Files to turn in: `ft_print_alphabet.c`	
Allowed functions: `write`	

- Create a function that displays the alphabet in lowercase, on a single line, in ascending order, starting from the letter 'a'.
- The function should be prototyped as follows:

```
void ft_print_alphabet(void);
```

Don't hesitate to randomly ask someone in your cluster if you have a question.

Chapter VI

Exercise 02: ft_print_reverse_alphabet

	Exercise 02
ft_print_reverse_alphabet	
Turn-in directory: *ex*02/	
Files to turn in: `ft_print_reverse_alphabet.c`	
Allowed functions: `write`	

- Create a function that displays the alphabet in lowercase, on a single line, in descending order, starting from the letter 'z'.
- The function should be prototyped as follows:

```
void ft_print_reverse_alphabet(void);
```

Push your code to Git regularly!

Chapter VII

Exercise 03: ft_print_numbers

	Exercise 03
ft_print_numbers	
Turn-in directory: *ex*03/	
Files to turn in: `ft_print_numbers.c`	
Allowed functions: `write`	

- Create a function that displays all digits on a single line, in ascending order.
- The function should be prototyped as follows:

```
void ft_print_numbers(void);
```

Collaboration is the key to success.

Chapter VIII

Exercise 04: ft_is_negative

	Exercise 04
ft_is_negative	
Turn-in directory: *ex*04/	
Files to turn in: `ft_is_negative.c`	
Allowed functions: `write`	

- Create a function that displays 'N' or 'P' depending on the sign of the integer passed as a parameter.
  - If **n** is negative, display 'N'.
  - If **n** is positive or zero, display 'P'.
- The function should be prototyped as follows:

```
void ft_is_negative(int n);
```

`Failure is part of your learning journey.`

Milestone Achieved, Keep Going!

You have reached the end of the mandatory exercises required to validate this project.

Now, it's up to you to decide whether to continue with the following optional exercises or move on to your next project. Both paths will expose you to valuable concepts sooner or later.

To make your decision, consider the following points:

- The very first exam focuses on C programming. If you've already worked on the first C project, this experience will be useful. The same applies to the "rush" at the end of the week (you'll soon learn more about it).
- Your excellence in this Piscine is evaluated based on multiple factors. Completing each project is important, but your overall progress across all Piscine projects also plays a key role. Choose wisely to optimize your results.
- You can always revisit and retry the same project in a few days or weeks, up until the end of the Piscine.
- Staying synchronized with your peers fosters better collaboration.

Chapter IX

Exercise 05: ft_print_comb

	Exercise 05
ft_print_comb	
Turn-in directory: *ex*05/	
Files to turn in: `ft_print_comb.c`	
Allowed functions: `write`	

- Create a function that displays all unique combinations of three distinct digits, with both the digits within each combination and the combinations themselves in ascending order.

- Expected output:

```
$>./a.out | cat -e
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789$>
```

- **987** is not included because **789** already covers that combination.
- **999** is not included because the digit **9** appears more than once.
- The function should be prototyped as follows:

```
void ft_print_comb(void);
```

Did you check with your neighbor on the right?

Chapter X

Exercise 06: ft_print_comb2

	Exercise 06
	ft_print_comb2
Turn-in directory: *ex*06/	
Files to turn in: `ft_print_comb2.c`	
Allowed functions: `write`	

- Create a function that displays all different combinations of two distinct two-digits numbers (XX XX) between **00** and **99**, listed in ascending order.
- Expected output:

```
$>./a.out | cat -e
00 01, 00 02, 00 03, 00 04, 00 05, ..., 00 99, 01 02, ..., 97 99, 98 99$>
```

- The function should be prototyped as follows:

```
void ft_print_comb2(void);
```

Get inspired by others, but don't let them do the work for you!

Chapter XI

Exercise 07: ft_putnbr

	Exercise 07
ft_putnbr	
Turn-in directory: *ex*07/	
Files to turn in: `ft_putnbr.c`	
Allowed functions: `write`	

- Create a function that displays the number passed as a parameter. The function must be able to display all possible values of an **int** type variable.
- The function should be prototyped as follows:

```
void ft_putnbr(int nb);
```

- Example:
  - **ft_putnbr(42);** should display **42**.

Don't trust any single source of information, always run your own tests, checks, and verifications!

Chapter XII

Exercise 08: ft_print_combn

	Exercise 08
	ft_print_combn
Turn-in directory: *ex*08/	
Files to turn in: `ft_print_combn.c`	
Allowed functions: `write`	

- Create a function that displays all unique combinations of **n** distinct digits in ascending order, without repetition.
- The value of **n** will be such that: $0 < \mathbf{n} < 10$.
- Example output for $\mathbf{n} = 2$:

```
$>./a.out | cat -e
01, 02, 03, ..., 09, 12, ..., 79, 89$>
```

- The function should be prototyped as follows:

```
void ft_print_combn(int n);
```

Did you check with your neighbor on the left?

Chapter XIII

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. Make sure to double-check the names of your files to ensure they are correct.

`You must submit only the files specified in the project instructions.`