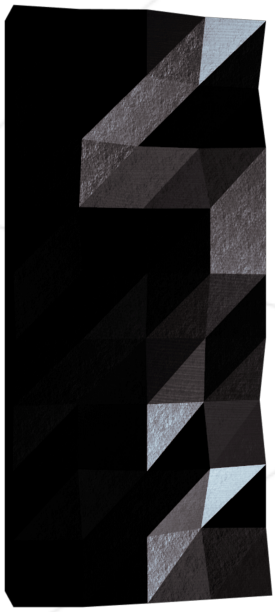

C Piscine

C 04

Summary: this document is the subject for the C 04 module of the C Piscine @ 42.

Version: 5

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 launched 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

Here are the lyrics for `City Hunter`'s theme song "Moonlight Shadow":

```
The last time ever she saw him
Carried away by a moonlight shadow
He passed on worried and warning
Carried away by a moonlight shadow.
Lost in a riddle that Saturday night
Far away on the other side.
He was caught in the middle of a desperate fight
And she couldn't find how to push through

The trees that whisper in the evening
Carried away by a moonlight shadow
Sing a song of sorrow and grieving
Carried away by a moonlight shadow
All she saw was a silhouette of a gun
Far away on the other side.
He was shot six times by a man on the run
And she couldn't find how to push through

[Chorus]
I stay, I pray
See you in Heaven far away...
I stay, I pray
See you in Heaven one day.

Four A.M. in the morning
Carried away by a moonlight shadow
I watched your vision forming
Carried away by a moonlight shadow
A star was glowing in the silvery night
Far away on the other side
Will you come to talk to me this night
But she couldn't find how to push through

[Chorus]
```

```
Far away on the other side.
Caught in the middle of a hundred and five
The night was heavy and the air was alive
But she couldn't find how to push through
Carried away by a moonlight shadow
Carried away by a moonlight shadow
Far away on the other side.
```

Unfortunately, this module has nothing to do with `City Hunter`.

Chapter IV

Exercise 00 : ft_strlen

	Exercise 00
ft_strlen	
Turn-in directory: $ex00/$	
Files to turn in: `ft_strlen.c`	
Allowed functions: `None`	

- Create a function that counts and returns the number of characters in a string.
- The function should be prototyped as follows:

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

Chapter V

Exercise 01 : ft_putstr

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

- Create a function that prints a string of characters to the standard output.
- The function should be prototyped as follows:

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

Chapter VI

Exercise 02 : ft_putnbr

	Exercise 02
ft_putnbr	
Turn-in directory: *ex*02/	
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 handle all possible values of the **int** type.
- The function should be prototyped as follows:

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

- Example usage:
  - `ft_putnbr(42)` outputs "42".

Chapter VII

Exercise 03 : ft_atoi

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

- Write a function that converts the initial portion of the string pointed to by `str` into its integer representation.
- The string may begin with an arbitrary amount of whitespace (as determined by `isspace(3)`).
- The string may be preceded by an arbitrary number of `'+'` and `'-'` signs. A `'-'` sign will invert the result depending on whether the number of `'-'` signs is odd or even.
- The function should then process any consecutive digits in base 10.
- The function reads the string until a non-conforming character is encountered and returns the number obtained so far.
- Overflow and underflow do not need to be handled; the function's return value is undefined in such cases.
- **Example Program Output**:

```
$>./a.out "   ---+--+1234ab567"
-1234
```

- The function should be prototyped as follows:

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

Chapter VIII

Exercise 04 : ft_putnbr_base

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

- Create a function that displays a number in a given base system in the terminal.
- The number is provided as an **int**, and the base is represented as a **string of characters**.
- The base system consists of all the symbols used to represent the number.
  - `0123456789` is the commonly used base system for representing decimal numbers.
  - `01` is a binary base system.
  - `0123456789ABCDEF` is a hexadecimal base system.
  - `poneyvif` is an octal base system.
- The function must handle negative numbers.
- If an invalid argument is provided, nothing should be displayed.
  Examples of invalid arguments:
  - The base is empty or has only one character.
  - The base contains duplicate characters.
  - The base contains '`+`' or '`-`'.

- The function should be prototyped as follows:

```
void        ft_putnbr_base(int nbr, char *base);
```

Chapter IX

Exercise 05 : ft_atoi_base

	Exercise 05
ft_atoi_base	
Turn-in directory: *ex*05/	
Files to turn in: `ft_atoi_base.c`	
Allowed functions: `None`	

- Write a function that converts the initial portion of the string pointed to by `str` into an integer representation.
- `str` is in a specific base, given as a second parameter.
- Except for the base rule, the function should behave exactly like **ft_atoi**.
- If an invalid argument is provided, the function should return 0.
  Examples of invalid arguments:
  - The base is empty or has only one character.
  - The base contains duplicate characters.
  - The base contains `'+'`, `'-'`, or whitespace characters.
- The function should be prototyped as follows:

```
int        ft_atoi_base(char *str, char *base);
```

Chapter X

Submission and peer-evaluation

Submit your assignment to your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the filenames to ensure they are correct.

`You must submit only the files required by the project instructions.`