



PISCINE

< INTRO TO C & POINTERS />



PISCINE

Preliminaries



Language: C

All of your source files, except for useless files (binary, temp files, obj files,...), must be included in your delivery.



- Don't push your `main` function into your delivery directory, we will be adding our own. Your files will be compiled adding our `main.c` and our `my_putchar.c` files.
- You are only allowed to use the `my_putchar` function to complete the following tasks, but **don't push it** into your delivery directory, and don't copy it in *any* of your delivered files.
- If one of your files prevents you from compiling with `*.c`, the Autograder will not be able to correct your work and you will receive a 0.



The use of AI is strictly forbidden (ChatGPT, Claude, Copilot, Gemini or any other assistant).

The goal of the day is to **search and try by yourself**: read the subject and the error messages, use `man`, test, ask your peers, then call a member of the pedagogical team.



Clone your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.
The Piscine runs from **10:00 to 18:00**: make sure your last push is done before 18:00.

PISCINE

Grading

The day is split into two parts. **Part 1** covers the fundamentals: completing all of its tasks gets you to **50%** of the grade. **Part 2** is more complex and makes up the remaining **50%**. Do the tasks in order: each one prepares the next.

Part	Task	Points
Part 1	Task 01 - my_print_alpha	7
Part 1	Task 02 - my_print_digits	7
Part 1	Task 03 - my_isneg	8
Part 1	Task 04 - my_swap	9
Part 1	Task 05 - my_putstr	9
Part 1	Task 06 - my_strlen	10
Fundamentals	Subtotal	50
Part 2	Task 07 - my_print_comb	12
Part 2	Task 08 - my_put_nbr	12
Part 2	Task 09 - my_evil_str	12
Part 2	Task 10 - my_getnbr	14
Going further	Subtotal	50
Total		100

Task 00 - Coding Style

At Epitech, every C code must comply with our Coding Style: **Epitech C Coding Style**.



Read it carefully! The code quality is an important factor and will be evaluated!

In order to check if your project is compliant with the coding style (for the most part), you can use the `epiclang` tool. Using it is very simple, after ending the Task 01, open a terminal in your repository and run `epiclang my_print_alpha.c`: the tool will report any issues with your code.

PART 1 - FUNDAMENTALS

These tasks cover the basics of C: displaying characters, then your first pointers. Completing all of them correctly gets you to **50%** of the grade.



Arrays are **forbidden** for Tasks 01 to 03.

Task 01 - my_print_alpha

7 pts

Delivery: my_print_alpha.c

Write a function that, beginning with **a**, displays the lowercase alphabet in ascending order, on a single line. It must be prototyped as follows:

```
int my_print_alpha(void);
```

Task 02 - my_print_digits

7 pts

Delivery: my_print_digits.c

Write a function that displays all the digits, on a single line, in ascending order. It must be prototyped as follows:

```
int my_print_digits(void);
```

Task 03 - my_isneg

8 pts

Delivery: my_isneg.c

Write a function that displays either **N** if the integer passed as parameter is negative or **P**, if positive or null. It must be prototyped as follows:

```
int my_isneg(int n);
```

Task 04 - my_swap

9 pts

Delivery: my_swap.c

Write a function that swaps the content of two integers, whose addresses are given as parameters. It must be prototyped as follows:

```
void my_swap(int *a, int *b);
```

Task 05 - my_putstr

9 pts

Delivery: my_putstr.c

Write a function that displays, one-by-one, the characters of a string.

The address of the string's first character will be found in the pointer passed as a parameter to the function, which must be prototyped as follows:

```
int my_putstr(char const *str);
```

Task 06 - my_strlen

10 pts

Delivery: my_strlen.c

Write a function that counts and returns the number of characters found in the string passed as a parameter. It must be prototyped as follows:

```
int my_strlen(char const *str);
```

PART 2 - GOING FURTHER

These tasks are more complex: they make up the remaining **50%** of the grade. Take your time, test your functions, and don't give up!



Testing is an important part of software development. Create a `main` function in a separate file to check each of your functions separately, covering both regular and irregular situations. Don't push it!

Task 07 - my_print_comb

12 pts

Delivery: my_print_comb.c

Write a function that displays, in ascending order, all the numbers composed by three **different** digits numbers (012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789). Given three digits (all different), only the smallest number composed by those digits must be displayed. It must be prototyped as follows:

```
int my_print_comb(void);
```



Terminal



```
~$> ./a.out  
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789
```



Neither 987 nor 999 is to be displayed (as an example).

Task 08 - my_put_nbr

12 pts

Delivery: my_put_nbr.c

Write a function that displays the number given as a parameter. It must be able to display **all** the possible values of an int, and must be prototyped as follows:

```
int my_put_nbr(int nb);
```



For instance, my_put_nbr(42) displays 42, my_put_nbr(0) displays 0, my_put_nbr(-2147483647) displays -2147483647.

Delivery: my_evil_str.c

The goal of this task is to swap each of the string's characters, two by two. In other words, you will swap the first letter with the last one, the second with the second-to-last and so on.

The function should return a pointer to the first character of the reversed string:

```
char *my_evil_str(char *str);
```

For instance

```
a => a  
ab => ba  
abc => cba  
abcd => dcba  
abcde => edcba  
abcdef => fedcba
```



When testing your function you may encounter “Segmentation fault” errors. Either you’re messing with the pointers in your function or the string given in parameter is read-only!



Easy way to have read/write string for testing purpose: `man 3 strdup`.

Task 10 - my_getnbr

14 pts

Delivery: my_getnbr.c

Write a function that returns a number, sent to the function as a string. It must be prototyped as follows:

```
int my_getnbr(char const *str);
```

Here are some tricky examples to be handled:

```
"+-----+-----+-----+42" => -42
```

```
"42a43" => 42
```

```
"110000000000000000000000000042" => 0 (the number doesn't fit in an integer)
```

```
"-100000000000000000000000000042" => 0 (for the same reason)
```

{EPITECH}