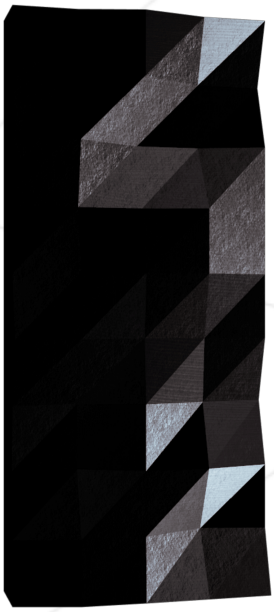


# C Piscine

## C 01

*Summary: This document is the subject for the module C 01 of the C Piscine @ 42.*

*Version: 5.4*

# Contents

# Chapter I

# Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- You'll only have to submit a main() function if we ask for a program.
- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses `cc`.
- If your program doesn't compile, you'll get `0`.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.

- Your reference guide is called `Google / man / the Internet / ....`
- Check out the "C Piscine" part of the forum on the intranet, or the slack Piscine.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!

Do not forget to add the *standard 42 header* in each of your .c/.h files. The norminette check its existence anyway!

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

# Chapter II

# Foreword

```
Vincent: And you know what they call a... a... a Quarter Pounder with Cheese
in Paris?

Jules: They don't call it a Quarter Pounder with cheese?

Vincent: No man, they got the metric system. They wouldn't know what the fuck
a Quarter Pounder is.

Jules: Then what do they call it?

Vincent: They call it a Royale with cheese.

Jules: A Royale with cheese. What do they call a Big Mac?

Vincent: Well, a Big Mac's a Big Mac, but they call it le Big-Mac.

Jules: Le Big-Mac. Ha ha ha ha. What do they call a Whopper?

Vincent: I dunno, I didn't go into Burger King.
```

At least one of the following exercices has nothing to do you with a Royale with cheese.

# Today's threshold

The validation threshold for this project is 50%.

It is up to you to determine which exercise allows you to reach this threshold, and if you want to complete more exercises.

# Chapter III

# Exercise 00 : ft_ft

| | Exercise 00 |
|---|---|
| | ft_ft |
| Turn-in directory : *ex*00/ | |
| Files to turn in : `ft_ft.c` | |
| Allowed functions : `None` | |

- Create a function that takes a pointer to int as a parameter, and sets the value "42" to that int.
- Here's how it should be prototyped :

```
void        ft_ft(int *nbr);
```

```
void ft_ft(int *nbr)
    *nbr = 42;

int main(void)
{
    int i;
    ft_ft(&i);
    return (0);
}
```

# Chapter IV

# Exercise 01 : ft_ultimate_ft

| | |
|---|---|
| | Exercise 01 |
| ft_ultimate_ft | |
| Turn-in directory : $ex01/$ | |
| Files to turn in : `ft_ultimate_ft.c` | |
| Allowed functions : `None` | |

- Create a function that takes a pointer to pointer to pointer to pointer to pointer to pointer to pointer to pointer to pointer to int as a parameter and sets the value "42" to that int.
- Here's how it should be prototyped :

```
void        ft_ultimate_ft(int *********nbr);
```

# Chapter V

# Exercise 02 : ft_swap

| | Exercise 02 |
|---|---|
| a | ft_swap |
| b | Turn-in directory : $ex02/$ |
| | Files to turn in : `ft_swap.c` |
| | Allowed functions : `None` |

- Create a function that swaps the value of two integers whose addresses are entered as parameters.
- Here's how it should be prototyped :

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

```
void ft-swap(int *a, int *b)
{
    int c;
    c = *a;
    *a = *b
    *b = c;
}

int main (void)
{
```

# Chapter VI

# Exercise 03 : ft_div_mod

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

- Create a function `ft_div_mod` prototyped like this :

```
void    ft_div_mod(int a, int b, int *div, int *mod);
```

- This function divides parameters `a` by `b` and stores the result in the int pointed by `div`. It also stores the remainder of the division of `a` by `b` in the int pointed by `mod`.

# Chapter VII

# Exercise 04 : ft_ultimate_div_mod

| | |
|---|---|
| | Exercise 04 |
| ft_ultimate_div_mod | |
| Turn-in directory : $ex04/$ | |
| Files to turn in : `ft_ultimate_div_mod.c` | |
| Allowed functions : `None` | |

- Create a function `ft_ultimate_div_mod` with the following prototype :

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

- This function divides parameters `a` by `b`. The result of this division is stored in the int pointed by `a`. The remainder of the division is stored in the int pointed by `b`.

# Chapter VIII

# Exercise 05 : ft_putstr

| | |
|---|---|
| | Exercise 05 |
| ft_putstr | |
| Turn-in directory : $ex05/$ | |
| Files to turn in : `ft_putstr.c` | |
| Allowed functions : `write` | |

- Create a function that displays a string of characters on the standard output.
- Here's how it should be prototyped :

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

# Chapter IX

# Exercise 06 : ft_strlen

| | Exercise 06 |
|---|---|
| | ft_strlen |
| Turn-in directory : *ex06/* | |
| Files to turn in : `ft_strlen.c` | |
| Allowed functions : `None` | |

- Create a function that counts and returns the number of characters in a string.
- Here's how it should be prototyped :

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

```
int ft_strlen(char *str)
{
    int strlen;
    strlen = 0;
    while (*str != 0)
    {
        strlen++;
        str++;
    }
    return (strlen);
}
```

# Chapter X

# Exercise 07 : ft_rev_int_tab

| | |
|---|---|
| | Exercise 07 |
| | ft_rev_int_tab |
| Turn-in directory : *ex*07/ | |
| Files to turn in : `ft_rev_int_tab.c` | |
| Allowed functions : `None` | |

0 1 2 3

int tab[] ; tab = {1 2 3 4} ;

- Create a function which reverses a given array of integer (first goes last, etc).
- The arguments are a pointer to int and the number of ints in the array.
- Here's how it should be prototyped :

0 4 8 12
90 94 98 102

```
void    ft_rev_int_tab(int *tab, int size);
```

[0, 1, 2, 3]

```
ft_rev_int_tab(int *tab, int size)
{
  int s;
  s = size;
  int rev_tab[];
  while ( s )
  { s --;
  *rev_tab = *tab + s (4);
   rev_tab ++;
  }
  int i;
  i = 0;
  while (i < size)
   { *tab = *rev_tab;
     tab ++;
     i++;
  }
```

no size in here

overcomplicated

you learned about address calculation

```
void ft_swap(int *tab, int size)
{
    int aux = 0;
    int *max = tab + size - 1;
    while ( tab <= max )
    {
        aux = *tab;
        *tab = *max;
        *max = aux;
        tab++;
        max--;
    }
}
```

# Chapter XI

# Exercise 08 : ft_sort_int_tab

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

- Create a function which sorts an array of integers by ascending order.
- The arguments are a pointer to int and the number of ints in the array.
- Here's how it should be prototyped :

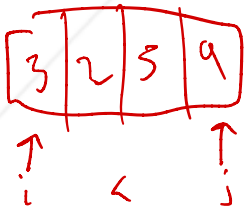
```
void    ft_sort_int_tab(int *tab, int size);
```

0 4 8 12
90 94 98 102

[0, 1, 2, 3]

9, 5, 3, 2

```
ft_sort_int_tab(int *tab, int size)
{
    int max = *tab
    int *last = tab + size - 1;
    while ( tab <= last )
    {
        if (*tab > max)
            max = *tab;
        tab++;
    }
}
```




swap
j--;
if (i == j)
i++;
last reduction

int = 0
i = 0(10) + *str - '0'
i = 0 + 1 - 48
i = 1

i = 1
i = 1(10) + 0 - '0'
i = 10 + 0
i = 10

i = 10
i = 10(10)+3
i = 100+3
i = 103

# Chapter XII

# Submission and peer-evaluation

Turn in 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 need to return only the files requested by the subject of this project.`