



C Piscine

C 11

Summary: This document is the subject for the module C 11 of the C Piscine @ 42.

Version: 7

Contents

| | | |
|-------------|--|-----------|
| I | Instructions | 2 |
| II | Foreword | 4 |
| III | Exercise 00 : ft_foreach | 6 |
| IV | Exercise 01 : ft_map | 7 |
| V | Exercise 02 : ft_any | 8 |
| VI | Exercise 03 : ft_count_if | 9 |
| VII | Exercise 04 : ft_is_sort | 10 |
| VIII | Exercise 05 : do-op | 11 |
| IX | Exercise 06 : ft_sort_string_tab | 13 |
| X | Exercise 07 : ft_advanced_sort_string_tab | 14 |
| XI | Submission and peer-evaluation | 15 |

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!!!

Chapter II

Foreword

Here's a little story :

(1982, California) Larry Walters of Los Angeles is one of the few to contend for the Darwin Awards and live to tell the tale. "I have fulfilled my 20-year dream," said Walters, a former truck driver for a company that makes TV commercials. "I'm staying on the ground. I've proved the thing works."

Larry's boyhood dream was to fly. But fates conspired to keep him from his dream. He joined the Air Force, but his poor eyesight disqualified him from the job of pilot. After he was discharged from the military, he sat in his backyard watching jets fly overhead.

He hatched his weather balloon scheme while sitting outside in his "extremely comfortable" Sears lawnchair. He purchased 45 weather balloons from an Army-Navy surplus store, tied them to his tethered lawnchair (dubbed the Inspiration I) and filled the four-foot diameter balloons with helium. Then, armed with some sandwiches, Miller Lite, and a pellet gun, he strapped himself into his lawnchair. He figured he would shoot to pop a few of the many balloons when it was time to descend.

Larry planned to sever the anchor and lazily float to a height of about 30 feet above the backyard, where he would enjoy a few hours of flight before coming back down. But things didn't work out quite as Larry planned.

When his friends cut the cord anchoring the lawnchair to his Jeep, he did not float lazily up to 30 feet. Instead he streaked into the LA sky as if shot from a cannon, pulled by the lift of 45 helium balloons, holding 33 cubic feet of helium each.

He didn't level off at 100 feet, nor did he level off at 1000 feet. After climbing and climbing, he leveled off at 16,000 feet.

At that height he felt he couldn't risk shooting any of the balloons, lest he unbalance the load and really find himself in trouble. So he stayed there, drifting cold and frightened with his beer and sandwiches, for more than 14 hours. He crossed the primary approach corridor of LAX, where startled Trans World Airlines and Delta Airlines pilots radioed in reports

of the strange sight.


Eventually he gathered the nerve to shoot a few balloons, and slowly descended. The hanging tethers tangled and caught in a power line, blacking out a Long Beach neighborhood for 20 minutes. Larry climbed to safety, where he was arrested by waiting members of the LAPD. As he was led away in handcuffs, a reporter dispatched to cover the daring rescue asked him why he had done it. Larry replied nonchalantly, "A man can't just sit around."

The Federal Aviation Administration was not amused. Safety Inspector Neal Savoy said, "We know he broke some part of the Federal Aviation Act, and as soon as we decide which part it is, a charge will be filed."

The moral of this story is Larry Walters should have stay on his chair and learn C....

Chapter III

Exercise 00 : ft_foreach

| | |
|---|--|
|  | Exercise 00 |
| | ft_foreach |
| | Turn-in directory : <i>ex00/</i> |
| | Files to turn in : ft_foreach.c |
| | Allowed functions : None |

- Create the function **ft_foreach**, which, for a given integer array, applies a function to all elements of the array. This function will be applied in the order of the array.
- Here is how the function should be prototyped:


```
void ft_foreach(int *tab, int length, void(*f)(int));
```

- For example, the function **ft_foreach** could be called as follows to display all integers in the array:

```
ft_foreach(tab, 1337, &ft_putnbr);
```

Chapter IV

Exercise 01 : ft_map


| | |
|---|-------------|
|  | Exercise 01 |
| ft_map | |
| Turn-in directory : <i>ex01/</i> | |
| Files to turn in : ft_map.c | |
| Allowed functions : malloc | |

- Create the function **ft_map**, which, for a given integer array, applies a function to all elements of the array (in order) and returns an array containing all the return values.
- This function will be applied in the order of the array.
- Here is how the function should be prototyped:

```
int      *ft_map(int *tab, int length, int(*f)(int));
```


Chapter V

Exercise 02 : ft_any

| | |
|---|-------------|
|  | Exercise 02 |
| ft_any | |
| Turn-in directory : <i>ex02/</i> | |
| Files to turn in : ft_any.c | |
| Allowed functions : None | |


- Create a function **ft_any**, which returns 1 if, when passed to the function **f**, at least one element of the array returns a value other than 0. Otherwise, it should return 0.
- This function will be applied in the order of the array.
- Here is how the function should be prototyped:

```
int ft_any(char **tab, int(*f)(char*));
```

- The array will be delimited by a null pointer.

Chapter VI

Exercise 03 : ft_count_if


| | |
|---|---|
|  | Exercise 03 |
| | ft_count_if |
| | Turn-in directory : <i>ex03/</i> |
| | Files to turn in : ft_count_if.c |
| | Allowed functions : None |

- Create a function **ft_count_if**, which returns the number of elements in the array for which the function **f** does not return 0.
- This function will be applied in the order of the array.
- Here is how the function should be prototyped:

```
int ft_count_if(char **tab, int length, int(*f)(char*));
```

Chapter VII

Exercise 04 : ft_is_sort


| | |
|---|--|
|  | Exercise 04 |
| | ft_is_sort |
| | Turn-in directory : <i>ex04/</i> |
| | Files to turn in : ft_is_sort.c |
| | Allowed functions : None |

- Create a function `ft_is_sort`, which returns 1 if the array is sorted and 0 if it isn't.
- The function provided as an argument should return a negative integer if the first argument is less than the second, 0 if they are equal, or a positive integer for anything else.
- Here is how the function should be prototyped:

```
int ft_is_sort(int *tab, int length, int(*f)(int, int));
```

Chapter VIII

Exercise 05 : do-op

| | |
|---|-------------|
|  | Exercise 05 |
| do-op | |
| Turn-in directory : <i>ex05/</i> | |
| Files to turn in : Your program files | |
| Allowed functions : write | |

- Create a program called `do-op`.
- The program will be executed with three arguments: `do-op value1 operator value2`.
- Example:

```
$>./do-op 42 "+" 21
63
$>
```

- You should use an array of pointers to functions to handle the **operator**.
- In case of an invalid operator, your program should print 0.
- If the number of arguments is invalid, `do-op` should not display anything.
- Your program should accept and print the result for the following operators: `'+' '-' '/' '*'` and `'%'`.
- Your program should treat the values as integers.
- In case of division by 0, it should print:

```
Stop : division by zero
```

- In case of modulo by 0, it should print:


```
Stop : modulo by zero
```

- Here is an example of tests the Moulinette will run:

```
$> make clean
$> make
$> ./do-op
$> ./do-op 1 + 1
2
$> ./do-op 42amis - ---20toto12
62
$> ./do-op 1 p 1
0
$> ./do-op 1 + toto3
1
$>
$> ./do-op toto3 + 4
4
$> ./do-op foo plus bar
0
$> ./do-op 25 / 0
Stop : division by zero
$> ./do-op 25 % 0
Stop : modulo by zero
$>
```

Chapter IX

Exercise 06 : ft_sort_string_tab


| | |
|---|-------------|
|  | Exercise 06 |
| ft_sort_string_tab | |
| Turn-in directory : <i>ex06/</i> | |
| Files to turn in : ft_sort_string_tab.c | |
| Allowed functions : None | |

- Create the function `ft_sort_string_tab`, which sorts the strings in `tab` in ASCII order.
- `tab` will be null-terminated.
- The sorting will be performed by exchanging the array's pointers.
- Here is how it should be prototyped:

```
void ft_sort_string_tab(char **tab);
```

Chapter X

Exercise 07 : ft_advanced_sort_string_tab

| | |
|---|-------------|
|  | Exercise 07 |
| ft_advanced_sort_string_tab | |
| Turn-in directory : <i>ex07/</i> | |
| Files to turn in : ft_advanced_sort_string_tab.c | |
| Allowed functions : None | |

- Create the function `ft_advanced_sort_string_tab`, which sorts depending on the return value of the function provided as an argument.
- The sorting will be performed by exchanging the array's pointers.
- `tab` will be null-terminated.
- Here is how it should be prototyped:

```
void ft_advanced_sort_string_tab(char **tab, int(*cmp)(char *, char *));
```



Calling `ft_advanced_sort_string_tab()` with `ft_strcmp` as the second argument will return the same result as `ft_sort_string_tab()`.

Chapter XI

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.