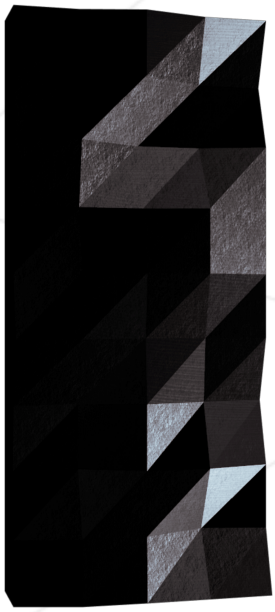

C Piscine

Day 10

Staff 42 pedago@42.fr

Summary: This document is the subject for Day10 of the C Piscine @ 42.

Contents

Chapter I

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- 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 `Norminator` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `Norminator`'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.
- If ft_putchar() is an authorized function, we will compile your code with our `ft_putchar.c`.
- 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 `gcc`.
- 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 your right. Otherwise, try your peer on your left.
- Your reference guide is called `Google / man / the Internet / ...`.
- Check out the "C Piscine" part of the forum on the intranet.
- 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 !!!

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 : Makefile

	Exercise 00
Makefile	
Turn-in directory : $ex00/$	
Files to turn in : `Makefile`	
Allowed functions : `None`	
Notes : `n/a`	

- Create the `Makefile` that'll compile your `libft.a`.
- The `Makefile` will get its source files from the "srcs" directory.
- The `Makefile` will get its header files from the "includes" directory.
- The lib will be at the root of the exercise.
- The `Makefile` should also implement the following rules: `clean`, `fclean` and `re` as well as `all`.
- `fclean` does the equivalent of a make clean and also erases the binary created during the make. `re` does the equivalent of a make fclean followed by a make.
- We'll only fetch your Makefile and test it with our files. For this exercise, only the following 5 mandatory functions of your lib have to be handled : (`ft_putchar`, `ft_putstr`, `ft_strcmp`, `ft_strlen` and `ft_swap`).

`Watch out for wildcards!`

Chapter IV

Exercise 01 : ft_foreach

	Exercise 01
ft_foreach	
Turn-in directory : $ex01/$	
Files to turn in : `ft_foreach.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create the function `ft_foreach` which, for a given ints array, applies a function on all elements of the array. This function will be applied following the array's order.
- Here's 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 in order to display all ints of the array :

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

Chapter V

Exercise 02 : ft_map

	Exercise 02
ft_map	
Turn-in directory : $ex02/$	
Files to turn in : `ft_map.c`	
Allowed functions : `malloc`	
Notes : `n/a`	

- Create the function `ft_map` which, for a given ints array, applies a function on all elements of the array (in order) and returns a array of all the return values. This function will be applied following the array's order.

- Here's how the function should be prototyped :

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

Chapter VI

Exercise 03 : ft_any

	Exercise 03
ft_any	
Turn-in directory : $ex03/$	
Files to turn in : `ft_any.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_any` which will return 1 if, passed to the function `f`, at least one element of the array returns 1. Else, it should return 0.
- Here's how the function should be prototyped :

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

- The array will be delimited by 0.

Chapter VII

Exercise 04 : ft_count_if

	Exercise 04
ft_count_if	
Turn-in directory : $ex04/$	
Files to turn in : `ft_count_if.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_count_if` which will return the number of elements of the array that return 1, passed to the function `f`.
- Here's how the function should be prototyped :

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

- The array will be delimited by 0.

Chapter VIII

Exercise 05 : ft_is_sort

	Exercise 05
ft_is_sort	
Turn-in directory : $ex05/$	
Files to turn in : `ft_is_sort.c`	
Allowed functions : `None`	
Notes : `n/a`	

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

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

Chapter IX

Exercise 06 : do-op

	Exercise 06
do-op	
Turn-in directory : $ex06/$	
Files to turn in : `Makefile, and your program files`	
Allowed functions : `write`	
Notes : `n/a`	

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

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

- The `operator` character corresponds to the appropriate function within an array of pointers to function.
- Your directory should contain a `Makefile` with the `all` and `clean` rules.
- In the case of an invalid argument such as `./do-op foo devide bar`, the program returns `0`.
- If the number of arguments is invalid, `do-op` doesn't display anything.

- Here's 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 X

Exercise 07 : ft_sort_wordtab

	Exercise 07
ft_sort_wordtab	
Turn-in directory : *ex*07/	
Files to turn in : `ft_sort_wordtab.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create the function `ft_sort_wordtab`, which sorts words obtained with `ft_split_whitespaces` by `ascii` order.
- The sorting will be performed by exchanging the array's pointers.
- Here's how it should be prototyped :

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

Chapter XI

Exercise 08 : ft_advanced_sort_wordtab

	Exercise 08
ft_advanced_sort_wordtab	
Turn-in directory : *ex*08/	
Files to turn in : `ft_advanced_sort_wordtab.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create the function `ft_advanced_sort_wordtab` which sorts, depending on the return of the function given as argument, words obtained with `ft_split_whitespaces`.
- The sorting will be performed by exchanging the array's pointers.
- Here's how it should be prototyped :

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

Calling ft_advanced_sort_wordtab() with ft_strcmp as a second argument will return the same result as ft_sort_wordtab().

Chapter XII

Exercise 09 : ft_advanced_do-op

	Exercise 09
ft_advanced_do-op	
Turn-in directory : $ex09/$	
Files to turn in : `Makefile, and your program files`	
Allowed functions : `write`	
Notes : `n/a`	

- Create a program that does the same as do-op with one difference : you have to include the file `ft_opp.h` which will define which pointer to function corresponds to which character.

```
$>cat ft_opp.h
#ifndef FT_OPP_H
# define FT_OPP_H

t_opp g_opptab[] =
{
    {"-", &ft_sub},
    {"+", &ft_add},
    {"*", &ft_mul},
    {"/", &ft_div},
    {"%", &ft_mod},
    {"", &ft_usage}
};

#endif
$>
```

- You'll have to create at least 6 functions: ft_add, ft_sub, ft_mul, ft_div, ft_mod, ft_usage.

- ft_usage displays the possible characters (defined in ft_opp.h) just like in the following example:

```
$> make clean
$> make
$> ./ft_advanced_do-op
$> ./ft_advanced_do-op 1 + 1
2
$> ./ft_advanced_do-op 1 p 1
error : only [ - + * / % ] are accepted.
$> ./ft_advanced_do-op 1 + toto3
1
$> ./ft_advanced_do-op 25 / 0
Stop : division by zero
$> ./ft_advanced_do-op 25 % 0
Stop : modulo by zero
$>
```

- You have to define the type of t_opp which corresponds to the s_opp structure allowing the compilation of your project.

- Don't write ANYTHING in the `ft_opp.h` file, not even t_opp's definition. Add the 42 header at the top of the file to respect the Norm. Include your own files if necessary.

- Only display an error for the operators that don't have a connection in `ft_opp.h`.

- We probably won't be using the same `ft_opp.h` every time...

`An operator can be made up of several characters.`