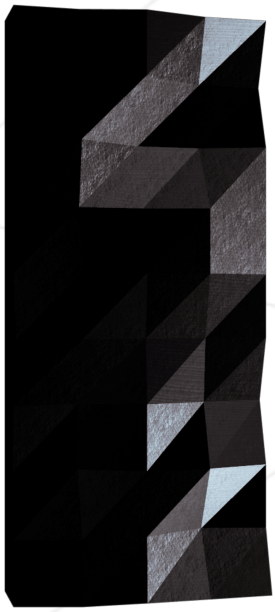

C Bootcamp

Day 03

Staff WeThinkCode_ info@wethinkcode.co.za

Summary: This document is the subject for Day03 of the C Bootcamp @ WeThinkCode_.

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 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 `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 a `-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 have to submit a main() function only 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 a `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 Bootcamp" 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 !!!

`Norminator must be launched with the -R CheckForbiddenSourceHeader flag. Moulinette will use it too.`

Chapter II

Foreword

"This life is what you make it. No matter what, you're going to mess up sometimes, it's a universal truth. But the good part is you get to decide how you're going to mess it up. Girls will be your friends - they'll act like it anyway. But just remember, some come, some go. The ones that stay with you through everything - they're your true best friends. Don't let go of them. Also remember, sisters make the best friends in the world. As for lovers, well, they'll come and go too. And baby, I hate to say it, most of them - actually pretty much all of them are going to break your heart, but you can't give up because if you give up, you'll never find your soulmate. You'll never find that half who makes you whole and that goes for everything. Just because you fail once, doesn't mean you're gonna fail at everything. Keep trying, hold on, and always, always, always believe in yourself, because if you don't, then who will, sweetie? So keep your head high, keep your chin up, and most importantly, keep smiling, because life's a beautiful thing and there's so much to smile about."

Marilyn Monroe

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`	
Notes : `n/a`	

- 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);
```

Chapter IV

Exercise 01 : ft_ultimate_ft

	Exercise 01
ft_ultimate_ft	
Turn-in directory : *ex*01/	
Files to turn in : `ft_ultimate_ft.c`	
Allowed functions : `None`	
Notes : `n/a`	

- 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
ft_swap	
Turn-in directory : *ex*02/	
Files to turn in : `ft_swap.c`	
Allowed functions : `None`	
Notes : `n/a`	

- 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);
```

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`	
Notes : `n/a`	

- 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 : *ex*04/	
Files to turn in : `ft_ultimate_div_mod.c`	
Allowed functions : `None`	
Notes : `n/a`	

- 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 : `ft_putchar`	
Notes : `n/a`	

- 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 : *ex*06/	
Files to turn in : `ft_strlen.c`	
Allowed functions : `None`	
Notes : `n/a`	

- 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);
```

Chapter X

Exercise 07 : ft_strrev

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

- Create a function that reverses the order of characters in a string.
- It has to return str.
- Here's how it should be prototyped :

```
char        *ft_strrev(char *str);
```

- Example:

```
a => a
ab => ba
abcde => edcba
```

Chapter XI

Exercise 08 : ft_atoi

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

- Reproduce the behavior of the function `atoi` (man atoi).
- Here's how it should be prototyped :

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

Chapter XII

Exercise 09 : ft_sort_integer_table

	Exercise 09
ft_sort_integer_table	
Turn-in directory : *ex*09/	
Files to turn in : `ft_sort_integer_table.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function which sorts an array (table) 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_integer_table(int *tab, int size);
```