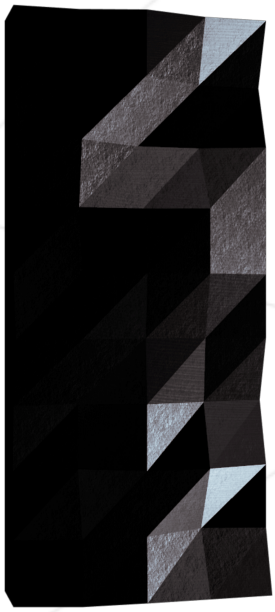

C Piscine

C 02

Resumen: Este documento corresponde al enunciado del proyecto C 02 de la C Piscine de 42.

Versión: 4

Índice general

Capítulo I

Instrucciones

- Esta página será la única referencia: no te fíes de los rumores.
- ¡Ten cuidado! Los enunciados pueden cambiar en cualquier momento.
- Asegúrate de que tus directorios y archivos tienen los permisos adecuados.
- Debes respetar el procedimiento de entrega para todos tus ejercicios.
- Tus compañeros de piscina se encargarán de corregir tus ejercicios.
- Además de por tus compañeros, también serán corregidos por un programa que se llama la Moulinette.
- La Moulinette es muy estricta a la hora de evaluar. Está completamente automatizada. Es imposible discutir con ella sobre tu nota. Por lo tanto, se extremadamente riguroso para evitar cualquier sorpresa.
- La Moulinette no tiene una mente muy abierta. No intenta comprender el código que no respeta la Norma. La Moulinette utiliza el programa `norminette` para comprobar La Norma en sus archivos. Entiende entonces que es estúpido entregar un código que no pase la `norminette`.
- Los ejercicios han sido ordenados con mucha precisión, del más sencillo al más complejo. En ningún caso se tendrá en cuenta un ejercicio complejo si no se ha conseguido realizar perfectamente un ejercicio más sencillo.
- El uso de una función prohibida se considera una trampa. Cualquier trampa será sancionada con la nota `-42`.
- Solamente hay que entregar una función main() si lo que se pide es un programa.
- La Moulinette compila con los flags -Wall -Wextra -Werror y utiliza `gcc`.
- Si tu programa no compila, tendrán un `0`.
- No puedes dejar en tu directorio ningún archivo que no se haya indicado de forma explícita en los enunciados de los ejercicios.
- ¿Tienes alguna pregunta? Pregunta a tu compañero de la derecha. Si no, prueba con tu compañero de la izquierda.

- Tu manual de referencia se llama `Google / man / Internet / ...`.
- ¡No olvides participar en el slack de tu Piscina!
- Lee detenidamente los ejemplos. Podrían exigir cosas que no se especifican necesariamente en los enunciados...
- Razona. ¡Te lo suplico, por Thor, por Odín! Maldita sea.

`Para este módulo, la Norminette debe ser ejecutada con el flag -R CheckForbiddenSourceHeader. La Moulinette también lo utilizará.`

Capítulo II

Introducción

He aquí un diálogo extraído de la serie `Silicon Valley`:

```
- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- Oh, God, help us! Okay, uh you know what? I just don't think this is
going to work. I'm so sorry. Uh, I mean like, what, we're going to bring
kids into this world with that over their heads? That's not really fair
to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no
way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SLAMS) - (BANGING)

. . .

(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.
```

Afortunadamente, no estás obligado a utilizar emacs y su barra espaciadora para completar los ejercicios siguientes.

Capítulo III

Ejercicio 00 : ft_strcpy

	Ejercicio: 00
ft_strcpy	
Directorio de entrega: *ex*00/	
Archivos a entregar: `ft_strcpy.c`	
Funciones autorizadas: `Ninguna`	

- Reproduce el comportamiento de la función `strcpy` (man strcpy)
- El prototipo de la función deberá ser el siguiente:

```
char        *ft_strcpy(char *dest, char *src);
```

Capítulo IV

Ejercicio 01 : ft_strncpy

	Ejercicio: 01
ft_strncpy	
Directorio de entrega: $ex01/$	
Archivos a entregar: `ft_strncpy.c`	
Funciones autorizadas: `Ninguna`	

- Reproduce el comportamiento de la función `strncpy` (man strncpy)
- El prototipo de la función deberá ser el siguiente:

```
char        *ft_strncpy(char *dest, char *src, unsigned int n);
```

Capítulo V

Ejercicio 02 : ft_str_is_alpha

	Ejercicio: 02
ft_str_is_alpha	
Directorio de entrega: $ex02/$	
Archivos a entregar: `ft_str_is_alpha.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que devuelva 1 si el string usado como parámetro contiene únicamente caracteres alfabéticos y devuelva 0 si contiene otro tipo de caracteres.
- El prototipo de la función deberá ser el siguiente:

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

- Tendrá que devolver 1 si `str` es un string vacío.

Capítulo VI

Ejercicio 03 : ft_str_is_numeric

	Ejercicio: 03
ft_str_is_numeric	
Directorio de entrega: *ex*03/	
Archivos a entregar: `ft_str_is_numeric.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que devuelva 1 si el string usado como parámetro contiene únicamente dígitos y devuelva 0 si contiene otro tipo de caracteres.
- El prototipo de la función deberá ser el siguiente:

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

- Tendrá que devolver 1 si `str` es un string vacío.

Capítulo VII

Ejercicio 04 : ft_str_is_lowercase

	Ejercicio: 04
ft_str_is_lowercase	
Directorio de entrega: $ex04/$	
Archivos a entregar: `ft_str_is_lowercase.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que devuelva 1 si el string usado como parámetro contiene únicamente caracteres alfabéticos en minúsculas y devuelva 0 si contiene otro tipo de caracteres.
- El prototipo de la función deberá ser el siguiente:

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

- Tendrá que devolver 1 si `str` es un string vacío.

Capítulo VIII

Ejercicio 05 : ft_str_is_uppercase

	Ejercicio: 05
ft_str_is_uppercase	
Directorio de entrega: $ex05/$	
Archivos a entregar: `ft_str_is_uppercase.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que devuelva 1 si el string usado como parámetro contiene únicamente caracteres alfabéticos en mayúsculas y devuelva 0 si contiene otro tipo de caracteres.
- El prototipo de la función deberá ser el siguiente:

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

- Tendrá que devolver 1 si `str` es un string vacío.

Capítulo IX

Ejercicio 06 : ft_str_is_printable

	Ejercicio: 06
ft_str_is_printable	
Directorio de entrega: $ex06/$	
Archivos a entregar: `ft_str_is_printable.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que devuelva 1 si el string usado como parámetro contiene únicamente caracteres imprimibles y devuelva 0 si contiene otro tipo de caracteres.
- El prototipo de la función deberá ser el siguiente:

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

- Tendrá que devolver 1 si `str` es un string vacío.

Capítulo X

Ejercicio 07 : ft_strupcase

	Ejercicio: 07
ft_strupcase	
Directorio de entrega: $ex07/$	
Archivos a entregar: `ft_strupcase.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que ponga cada letra en mayúscula.
- El prototipo de la función deberá ser el siguiente:

```
char        *ft_strupcase(char *str);
```

- Deberá devolver `str`.

Capítulo XI

Ejercicio 08 : ft_strlowcase

	Ejercicio: 08
ft_strlowcase	
Directorio de entrega: $ex08/$	
Archivos a entregar: `ft_strlowcase.c`	
Funciones autorizadas: `Ninguna`	

- Crea una función que ponga cada letra en minúscula.
- El prototipo de la función deberá ser el siguiente:

```
char        *ft_strlowcase(char *str);
```

- Deberá devolver `str`.

Capítulo XII

Ejercicio 09 : ft_strcapitalize

	Ejercicio: 09
ft_strcapitalize	
Directorio de entrega: *ex*09/	
Archivos a entregar: `ft_strcapitalize.c`	
Funciones autorizadas: `Ninguna`	

- Creaa una función que ponga en mayúscula la primera letra de cada palabra y el resto de la palabra en minúsculas.
- Una palabra es una secuencia (string) de caracteres alfanuméricos.
- El prototipo de la función deberá ser el siguiente:

```
char        *ft_strcapitalize(char *str);
```

- Deberá devolver `str`.
- Por ejemplo:

```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

- Se convierte en:

```
Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un
```

Capítulo XIII

Ejercicio 10 : ft_strlcpy

	Ejercicio: 10
ft_strlcpy	
Directorio de entrega: $ex10/$	
Archivos a entregar: `ft_strlcpy.c`	
Funciones autorizadas: `Ninguna`	

- Reproduce el comportamiento de la función `strlcpy` (man strlcpy)
- El prototipo de la función deberá ser el siguiente:

```
unsigned int ft_strlcpy(char *dest, char *src, unsigned int size);
```

Capítulo XIV

Ejercicio 11 : ft_putstr_non_printable

	Ejercicio: 11
ft_putstr_with_non_printable	
Directorio de entrega: $ex11/$	
Archivos a entregar: `ft_putstr_non_printable.c`	
Funciones autorizadas: `write`	

- Escribe una función que muestre una secuencia de caracteres en la pantalla. Si esta secuencia contiene caracteres no imprimibles, deberán ser mostrados en formato hexadecimal (en minúsculas) precedidos de una barra invertida (backslash).
- Por ejemplo, con este parámetro:

```
Coucou\ntu vas bien ?
```

- La función tendrá que mostrar:

```
Coucou\0atu vas bien ?
```

- El prototipo de la función deberá ser el siguiente:

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

Capítulo XV

Ejercicio 12 : ft_print_memory

	Ejercicio: 12
ft_print_memory	
Directorio de entrega: $ex12/$	
Archivos a entregar: `ft_print_memory.c`	
Funciones autorizadas: `write`	

- Escriba una función que muestre la región de memoria en la pantalla.
- La visualización de la región de memoria estará dividida en tres "columnas" separadas por un espacio:
  - La dirección en hexadecimal del primer carácter de la línea seguido de ":".
  - El contenido en hexadecimal, con un espacio cada dos caracteres, deberá ser completado con espacios si es preciso (ver el ejemplo a continuación).
  - El contenido en caracteres imprimibles.
- Si un carácter es no imprimible será remplazado por un punto.
- Cada línea debe contener dieciséis caracteres.
- Si `size` es igual a 0, no se muestra nada.

- Ejemplo:

```
$> ./ft_print_memory
000000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 2020 6573 7420 666f ches...c est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010a161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.
000000010a161f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..
000000010a161f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000      ..lol.lol. .
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
0000000107ff9f50: 6368 6573 090a 0963 2020 6573 7420 666f ches...c est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.$
0000000107ff9f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..$
0000000107ff9f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000      ..lol.lol. .$
$>
```

- El prototipo de la función deberá ser el siguiente:

```
void        *ft_print_memory(void *addr, unsigned int size);
```

- Deberá devolver `addr`.