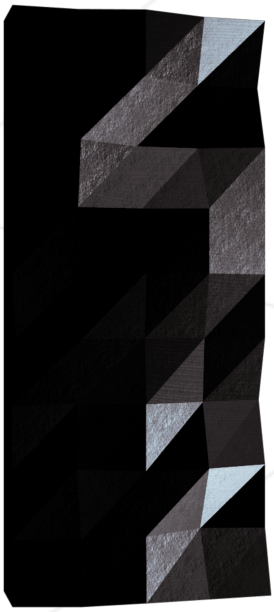

C Piscine

C 12

Resumen: Este documento corresponde al enunciado del módulo C 12 de la C Piscine de 42.

Versión: 9

Índice general

Capítulo I

Introducción

AVISO DE SPOILER
NO LEA LA PÁGINA SIGUIENTE

Tú lo has querido.

- En `Star Wars`, Darth Vader es el padre de Luke Skywalker.
- En `Sospechosos habituales`, Verbal es Keyser Soze.
- En `El club de la lucha`, Tyler Durden y el narrador son la misma persona.
- En `El sexto sentido`, Bruce Willis está muerto desde el principio.
- En `Los otros`, los residentes de la casa son los fantasmas y viceversa.
- En `Bambi`, muere la madre de Bambi.
- En `El bosque`, los monstruos son los aldeanos y, en realidad, la historia ocurre en nuestra época.
- En `Harry Potter`, muere Dumbledore.
- En `El planeta de los simios`, la historia se desarrolla en la Tierra.
- En `Juego de tronos`, Robb Stark y Joffrey Baratheon mueren en su noche de bodas.
- En `Crepúsculo`, los vampiros brillan cuando se exponen al sol.
- En `Stargate SG-1, Temporada 1, Episodio 18`, O'Neill y Carter están en la Antártida.
- En `Fullmetal Alchemist: Brotherhood`, Alfonse da su vida para que su hermano pueda derrotar a su padre, el auténtico villano.
- En `El caballero oscuro: la leyenda renace`, Miranda Tate es Talia al Ghul.
- En `Super Mario Bros`, la princesa se encuentra en otro castillo.
- En `Ataque a los Titanes`, después de que Eren provoque el rumbling y acabe con el 80 % de la población mundial, Mikasa lo mata para detenerlo

Capítulo II

Instrucciones

- Esta página será la única referencia: no te fíes de los rumores.
- Antes de empezar a hacer los ejercicios deberás registrarte en el proyecto en tu intranet. Una vez lo hayas hecho, obtendrás tu repositorio Git en el cual debes trabajar.
- Los ejercicios han sido ordenados del más sencillo al más complejo. Por lo tanto, debes hacerlos en orden y asegurarte de que el ejercicio es correcto antes de continuar con el siguiente.
- Los ejercicios de Shell se deben ejecutar con `/bin/bash`.
- Los ejercicios de C se deben compilar con `cc` y utilizando las flags -Wall -Wextra -Werror.
- Solamente hay que entregar una función `main()` si lo que se pide es un programa. Si se pide una función se puede entregar el `main()` comentado con la batería de tests que hayas hecho.
- Los ejercicios de C se deben escribir de acuerdo a **la Norma**. Puedes encontrarla en la intranet, en el icono de la brújula, FAQ, sección 42, apartado General Pedagogy como `La Norma`.
- Para ayudarte a comprobar si tus ejercicios cumplen **la Norma** puedes ayudarte de `norminette` - un programa que aplica ciertos requisitos de **la Norma** a tu código. Si tienes archivos o funciones adicionales, también deben cumplir con **la Norma**.
- Lee detenidamente los ejemplos. Podrían dar información que no se especifica en el enunciado...
- Asegúrate de que tus directorios y archivos tienen los permisos adecuados.
- Debes respetar **el procedimiento de entrega** para todos tus ejercicios. Solo el trabajo de tu repositorio `Git` será evaluado.
- **No puedes** dejar en tu directorio **ningún** archivo que no se haya indicado de forma explícita en los enunciados de los ejercicios.

- La evaluación de este proyecto consta de dos fases. Primero, las personas con las que compartes piscina se encargarán de evaluar tus ejercicios. A continuación, serán evaluados por un programa que se llama **la Moulinette**.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free) excepto en el caso de comportamientos indefinidos. Si esto sucede, tu proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- **La Moulinette** es muy estricta a la hora de evaluar y está completamente automatizada. Es imposible discutir con ella sobre tu nota. Por lo tanto, debes mantener el rigor en tu código para evitar cualquier sorpresa.
- **La Moulinette** utiliza `norminette` para comprobar tus ejercicios, aunque este programa no es exhaustivo en la verificacion de **la Norma**. Durante las evaluaciones por parte de las personas de tu Piscina se comprobará que **la Norma** es correcta. Si se incumple el proyecto tendrá una puntuación final de **0**, aunque `norminette` no indique ningún error.
- ¿Tienes alguna pregunta? Pregunta a la persona de tu derecha. Si no puede ayudarte, prueba con la persona de tu izquierda.
- ¡Por Loki, por Freyja! ¡¡Piensa fuerte!!
- Para los ejercicios de hoy, utilizaremos la estructura siguiente:

```
typedef struct          s_list
{
        struct s_list   *next;
        void            *data;
}                       t_list;
```

- Debes colocar esta estructura en un archivo `ft_list.h` y entregarlo en cada ejercicio.
- A partir del ejercicio 01 utilizaremos nuestro ft_create_elem, así que tenlo en cuenta (podría ser interesante tener tu prototipo en ft_list.h...).

Capítulo III

Instrucciones sobre la IA

Contexto

La Piscina de C es intensa. Es tu primer gran desafío en 42: una inmersión profunda en la resolución de problemas, la autonomía y la comunidad.

Durante esta etapa, tu objetivo principal es construir unas bases sólidas, a través del esfuerzo, la repetición y, sobre todo, mediante el **aprendizaje entre pares**.

En la era de la IA, los atajos son fáciles de encontrar. Sin embargo, es importante considerar si el uso que haces de la IA te está ayudando realmente a crecer, o si simplemente te está impidiendo desarrollar habilidades reales.

La Piscina también es una experiencia humana y, por ahora, nada puede reemplazar eso. Ni siquiera la IA.

Para obtener una visión más completa de nuestra postura sobre la IA como herramienta de aprendizaje, como parte del plan de estudios de TIC (Tecnologías de la Información y la Comunicación) y como una expectativa creciente en el mercado laboral, consulta las preguntas frecuentes FAQ disponibles en la intranet.

Mensaje principal

- Construir fundamentos sólidos sin atajos.
- Desarollar de forma real habilidades técnicas y transversales.
- Experimentar el aprendizaje entre pares de forma real, empezar a aprender a aprender y a resolver nuevos problemas.
- El proceso de aprendizaje es más importante que el resultado.
- Aprender sobre los riesgos asociados a la IA y desarrollar prácticas de control efectivas y medidas de mitigación para evitar errores comunes.

Reglas para la piscina:

- Aplica la lógica y el razonamiento a las tareas asignadas, especialmente antes de recurrir a la IA.
- No deberías pedir respuestas directas a la IA.
- Aprende sobre el enfoque global de 42 sobre la IA.

Resultados de esta etapa:

Durante esta etapa fundamental, obtendrás los siguientes resultados:

- Obtener Una base sólida técnica y en programación.
- Comprender por qué y cómo la IA puede ser peligrosa durante esta fase.

Comentarios y ejemplos:

- Si, sabemos que la IA existe. Y sí, puede resolver tus proyectos. Pero estás aquí para aprender, no para demostrar que la IA ha aprendido. No pierdas tu tiempo (o el nuestro) solo para demostrar que la IA puede resolver el problema dado. y que puede resolver tus proyectos, pero estás aquí para aprender. Te recomendamos que inviertas 26 días de aprendizaje de calidad para resolver los retos que hemos pensado para ti.
- Aprender en 42 no va sólo de saber la respuesta. Se trata de desarrollar la capacidad de encontrar una. La IA te da la respuesta directamente, pero eso te impide construir tu propio razonamiento. Y el razonamiento requiere tiempo, esfuerzo y conlleva fracasos. El camino hacia el éxito no debe ser fácil. no tiene nada que ver con saber la respuesta. En 42 aprendemos desarrollando la capacidad para encontrarla. La IA te dará la respuesta directa, lo que impide que desarrolles tu propio razonamiento. Razonar requiere tiempo, esfuerzo y equivocare muchas veces.
- Piensa que durante los exámenes, la IA no está disponible sin internet, sin smartphones, etc. Te darás cuenta rápidamente si has confiado demasiado en la IA durante tu proceso de aprendizaje. Recuerda que durante los exámenes no tendrás acceso a la IA ni a internet. Te vas a enfrentar a una hoja en blanco donde vas a tener que escribir tu propio código.
- El aprendizaje entre pares te expone a diferentes ideas y enfoques, mejorando tus habilidades interpersonales y tu capacidad de pensar de forma diferente. Eso es mucho más valioso que simplemente chatear con un bot. Así que, ¡que no te supere la timidez!: ¡habla, haz preguntas y aprende con el resto de la piscina!

- Sí, la IA formará parte del plan de estudios, tanto como herramienta de aprendizaje como tema en sí mismo. Incluso tendrás la oportunidad de crear tu propio software de IA. Para aprender más sobre nuestro enfoque progresivo, puedes consultar la documentación disponible en la intranet.

✓ Buenas prácticas:

> Me atasco en un nuevo concepto. Le pregunto a alguien cercano cómo lo ha abordado. Hablamos durante 10 minutos y, de repente, todo encaja. Lo entiendo. No entiendo algo concreto del proyecto y no sé cómo continuar. Le pregunto a otra persona de la piscina cómo lo ha abordado, hablamos sobre el tema y, si es necesario, incluso utilizamos otros métodos (papel y boli, dibujos, metáforas, etc.) hasta conseguir entenderlo.

✗ Mala práctica:

> Utilizo la IA en secreto, copio un código que parece correcto. Durante la evaluación entre pares, no puedo explicar nada. Suspendo. Durante el examen, sin IA, me vuelvo a atascar. Suspendo.

Capítulo IV

Ejercicio 00 : ft_create_elem

	Ejercicio: 00
ft_create_elem	
Directorio de entrega: *ex*00/	
Archivos a entregar: `ft_create_elem.c, ft_list.h`	
Funciones autorizadas: `malloc`	

- Crea la función `ft_create_elem` que cree un elemento nuevo de tipo `t_list`.
- Tendrá que asignar `data` al parámetro proporcionado y `next` a NULL.
- El prototipo de la función deberá ser el siguiente:

```
t_list        *ft_create_elem(void *data);
```

Capítulo V

Ejercicio 01 : ft_list_push_front

	Ejercicio: 01
ft_list_push_front	
Directorio de entrega: $ex01/$	
Archivos a entregar: `ft_list_push_front.c, ft_list.h`	
Funciones autorizadas: `ft_create_elem`	

- Crea la función `ft_list_push_front` que añada al principio de la lista un elemento nuevo de tipo `t_list`.
- Tendrá que asignar `data` al parámetro proporcionado.
- Actualizará, si es preciso, el puntero al principio de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void        ft_list_push_front(t_list **begin_list, void *data);
```

Capítulo VI

Ejercicio 02 : ft_list_size

	Ejercicio: 02
ft_list_size	
Directorio de entrega: $ex02/$	
Archivos a entregar: `ft_list_size.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_size` que devuelva el número de elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

```
int ft_list_size(t_list *begin_list);
```

Capítulo VII

Ejercicio 03 : ft_list_last

	Ejercicio: 03
ft_list_last	
Directorio de entrega: $ex03/$	
Archivos a entregar: `ft_list_last.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_last` que devuelva el último elemento de la lista.
- El prototipo de la función deberá ser el siguiente:

```
t_list *ft_list_last(t_list *begin_list);
```

Capítulo VIII

Ejercicio 04 : ft_list_push_back

	Ejercicio: 04
ft_list_push_back	
Directorio de entrega: $ex04/$	
Archivos a entregar: `ft_list_push_back.c, ft_list.h`	
Funciones autorizadas: `ft_create_elem`	

- Crea la función `ft_list_push_back` que añada al final de la lista un elemento nuevo de tipo `t_list`.
- Tendrá que asignar `data` al parámetro proporcionado.
- Actualizará, si es preciso, el puntero al principio de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void        ft_list_push_back(t_list **begin_list, void *data);
```

Capítulo IX

Ejercicio 05 : ft_list_push_strs

	Ejercicio: 05
ft_list_push_strs	
Directorio de entrega: *ex*05/	
Archivos a entregar: `ft_list_push_strs.c, ft_list.h`	
Funciones autorizadas: `ft_create_elem`	

- Crea la función `ft_list_push_strs` que cree una lista nueva e introduzca en ella las cadenas de caracteres apuntadas por los elementos de la tabla `strs`.
- `size` es el tamaño de `strs`
- El primer elemento de la tabla se encontrará al final de la lista.
- Se devolverá la dirección del primer elemento de la lista.
- El prototipo de la función deberá ser el siguiente:

```
t_list *ft_list_push_strs(int size, char **strs);
```

Capítulo X

Ejercicio 06 : ft_list_clear

	Ejercicio: 06
ft_list_clear	
Directorio de entrega: $ex06/$	
Archivos a entregar: `ft_list_clear.c, ft_list.h`	
Funciones autorizadas: `free`	

- Crea la función `ft_list_clear` que retire y libere todos los elementos de la lista.
- También tendrán que liberarse todos los `data` usando `free_fct`.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_clear(t_list *begin_list, void (*free_fct)(void *));
```

Capítulo XI

Ejercicio 07 : ft_list_at

	Ejercicio: 07
ft_list_at	
Directorio de entrega: *ex*07/	
Archivos a entregar: `ft_list_at.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_at` que devuelva el n-ésimo elemento de la lista, sabiendo que el primer elemento es el elemento 0.
- En caso de error, devolverá un puntero nulo.
- El prototipo de la función deberá ser el siguiente:

```
t_list *ft_list_at(t_list *begin_list, unsigned int nbr);
```

Capítulo XII

Ejercicio 08 : ft_list_reverse

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

- Crea la función `ft_list_reverse` que invierta el orden de los elementos de la lista. El valor de cada elemento debe mantenerse igual.
- Atención, en este ejercicio utilizaremos nuestro propio ft_list.h.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_reverse(t_list **begin_list);
```

Capítulo XIII

Ejercicio 09 : ft_list_foreach

	Ejercicio: 09
ft_list_foreach	
Directorio de entrega: $ex09/$	
Archivos a entregar: `ft_list_foreach.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_foreach` que aplique una función pasada como parámetro al valor incluido en cada elemento de la lista.
- Se debe aplicar `f` en el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

- La función apuntada por `f` será utilizada de la siguiente forma:

```
(*f)(list_ptr->data);
```

Capítulo XIV

Ejercicio 10 : ft_list_foreach_if

	Ejercicio: 10
ft_list_foreach_if	
Directorio de entrega: $ex10/$	
Archivos a entregar: `ft_list_foreach_if.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_foreach_if` que aplique una función pasada como parámetro al valor incluido en algunos elementos de la lista.
- Solo se aplicará `f` a los elementos que, al ser pasados como argumento a `cmp` con `data_ref`, hagan que `cmp` devuelva 0.
- Se debe aplicar `f` en el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void        ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void
                                                        *data_ref, int (*cmp)())
```

- Las funciones apuntadas por `f` y por `cmp` serán utilizadas de la siguiente forma:

```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```

`La función cmp podría ser, por ejemplo, ft_strcmp...`

Capítulo XV

Ejercicio 11 : ft_list_find

	Ejercicio: 11
ft_list_find	
Directorio de entrega: $ex11/$	
Archivos a entregar: `ft_list_find.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_find` que devuelva la dirección del primer elemento cuyos datos, cuando se los compare con `data_ref` usando `cmp`, hagan que `cmp` devuelva 0.
- El prototipo de la función deberá ser el siguiente:

```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

- La función apuntada por `cmp` será utilizada de la siguiente forma:

```
(*cmp)(list_ptr->data, data_ref);
```

Capítulo XVI

Ejercicio 12 : ft_list_remove_if

	Ejercicio: 12
ft_list_remove_if	
Directorio de entrega: $ex12/$	
Archivos a entregar: `ft_list_remove_if.c, ft_list.h`	
Funciones autorizadas: `free`	

- Crea la función `ft_list_remove_if` que borre de la lista todos los elementos cuyos datos, cuando se los compare con `data_ref` usando `cmp`, hagan que `cmp` devuelva 0.
- También tendrán que liberarse todos los `data` de un elemento que se tenga que borrar usando `free_fct`.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *));
```

- Las funciones apuntadas por `free_fct` y por `cmp` serán utilizadas de la siguiente forma:

```
(*cmp)(list_ptr->data, data_ref);
(*free_fct)(list_ptr->data);
```

Capítulo XVII

Ejercicio 13 : ft_list_merge

	Ejercicio: 13
ft_list_merge	
Directorio de entrega: $ex13/$	
Archivos a entregar: `ft_list_merge.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_merge` que coloque los elementos de una lista `begin2` al final de otra lista `begin1`.
- No se permite la creación de elementos.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_merge(t_list **begin_list1, t_list *begin_list2);
```

Capítulo XVIII

Ejercicio 14 : ft_list_sort

	Ejercicio: 14
ft_list_sort	
Directorio de entrega: $ex14/$	
Archivos a entregar: `ft_list_sort.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_sort` que ordene de forma creciente el contenido de la lista, comparando dos elementos mediante una función de comparación de datos de dos elementos.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

- La función apuntada por `cmp` será utilizada de la siguiente forma:

```
(*cmp)(list_ptr->data, other_list_ptr->data);
```

`La función cmp podría ser, por ejemplo, ft_strcmp.`

Capítulo XIX

Ejercicio 15 : ft_list_reverse_fun

	Ejercicio: 15
ft_list_reverse_fun	
Directorio de entrega: $ex15/$	
Archivos a entregar: `ft_list_reverse_fun.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_list_reverse_fun` que invierta el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_reverse_fun(t_list *begin_list);
```

Capítulo XX

Ejercicio 16 : ft_sorted_list_insert

	Ejercicio: 16
ft_sorted_list_insert	
Directorio de entrega: $ex16/$	
Archivos a entregar: `ft_sorted_list_insert.c, ft_list.h`	
Funciones autorizadas: `ft_create_elem`	

- Crea la función `ft_sorted_list_insert` que cree un elemento nuevo y lo inserte en una lista ordenada de tal modo que la lista quede en orden creciente.
- El prototipo de la función deberá ser el siguiente:

```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

- La función apuntada por `cmp` será utilizada de la siguiente forma:

```
(*cmp)(list_ptr->data, other_list_ptr->data);
```

Capítulo XXI

Ejercicio 17 : ft_sorted_list_merge

	Ejercicio: 17
ft_sorted_list_merge	
Directorio de entrega: $ex17/$	
Archivos a entregar: `ft_sorted_list_merge.c, ft_list.h`	
Funciones autorizadas: `Ninguna`	

- Crea la función `ft_sorted_list_merge` que integre los elementos de una lista ordenada `begin2` dentro de otra lista ordenada `begin1`, de tal modo que la lista `begin1` quede en orden creciente.
- El prototipo de la función deberá ser el siguiente:

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

- La función apuntada por `cmp` será utilizada de la siguiente forma:

```
(*cmp)(list_ptr->data, other_list_ptr->data);
```

Capítulo XXII

Entrega y evaluación

Entrega tu proyecto en tu repositorio `Git` como de costumbre. Solo el trabajo entregado en el repositorio será evaluado durante la defensa. No dudes en comprobar varias veces los nombres de los archivos para verificar que sean correctos.

Sólo necesitas entregar los archivos requeridos por el enunciado de este proyecto.