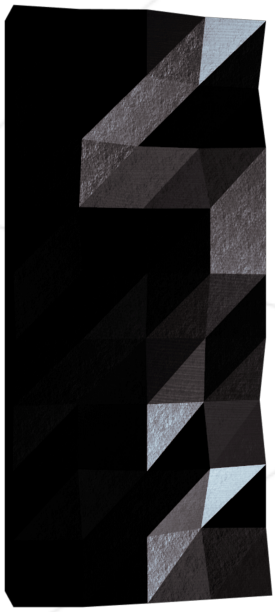

C Piscine

C 13

Resumen: Este documento corresponde al enunciado del C 13 de la C Piscine de 42.

Versión: 6

Índice general

Capítulo I

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 siguientes ejercicios, utilizaremos la siguiente estructura:

```
typedef struct          s_btree
{
        struct s_btree  *left;
        struct s_btree  *right;
        void            *item;
}                       t_btree;
```

- Deberás incluir esta estructura en un archivo llamado `ft_btree.h` y entregarlo dentro de cada ejercicio.
- A partir del ejercicio 01, utilizaremos nuestro `btree_create_node`, así que tenlo en cuenta.

`Para este último punto, tener el prototipo dentro del archivo ft_btree.h puede ser interesante...`

Capítulo II

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 III

Introducción

Aquí tienes una lista de publicaciones de `Venom`:

```
- In League with Satan (single, 1980)
- Welcome to Hell (1981)
- Black Metal (1982)
- Bloodlust (single, 1983)
- Die Hard (single, 1983)
- Warhead (single, 1984)
- At War with Satan (1984)
- Hell at Hammersmith (EP, 1985)
- American Assault (EP, 1985)
- Canadian Assault (EP, 1985)
- French Assault (EP, 1985)
- Japanese Assault (EP, 1985)
- Scandinavian Assault (EP, 1985)
- Manitou (single, 1985)
- Nightmare (single, 1985)
- Possessed (1985)
- German Assault (EP, 1987)
- Calm Before the Storm (1987)
- Prime Evil (1989)
- Tear Your Soul Apart (EP, 1990)
- Temples of Ice (1991)
- The Waste Lands (1992)
- Venom '96 (EP, 1996)
- Cast in Stone (1997)
- Resurrection (2000)
- Anti Christ (single, 2006)
- Metal Black (2006)
- Hell (2008)
- Fallen Angels (2011)
```

Este proyecto puede resultar más sencillo si lo haces mientras escuchas `Venom`.

Capítulo IV

Ejercicio 00 : btree_create_node

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

- Crea la función `btree_create_node` para crear un nuevo elemento. Deberá inicializar su `item` al valor del argumento, y el resto de elementos a `0`.
- Se devuelve la dirección al nodo creado.
- Aquí tienes el prototipo de la función:

```
t_btree *btree_create_node(void *item);
```

Capítulo V

Ejercicio 01 : btree_apply_prefix

	Ejercicio: 01
btree_apply_prefix	
Directorio de entrega: $ex01/$	
Archivos a entregar: `btree_apply_prefix.c, ft_btree.h`	
Funciones autorizadas: `Ninguna`	

- Crea una función `btree_apply_prefix` que ejecute la función dada como argumento al `item` de cada nodo, recorriéndolo de forma `preordinal`.
- Aquí tienes el prototipo:

```
void btree_apply_prefix(t_btree *root, void (*applyf)(void *));
```

Capítulo VI

Ejercicio 02 : btree_apply_infix

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

- Crea una función `btree_apply_infix` que aplique la función dada como segundo argumento al `item` de cada nodo, recorriendo el arbol de forma `inordinal`.
- Aquí tienes el prototipo:

```
void btree_apply_infix(t_btree *root, void (*applyf)(void *));
```

Capítulo VII

Ejercicio 03 : btree_apply_suffix

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

- Crea la función `btree_apply_suffix` que ejecute la función dada como segundo argumento sobre cada `item` de cada nodo, recorriendo el árbol de forma `postordinal`.
- Aquí tienes el prototipo:

```
void btree_apply_suffix(t_btree *root, void (*applyf)(void *));
```

Capítulo VIII

Ejercicio 04 : btree_insert_data

	Ejercicio: 04
btree_insert_data	
Directorio de entrega: *ex*04/	
Archivos a entregar: `btree_insert_data.c, ft_btree.h`	
Funciones autorizadas: `btree_create_node`	

- Crea una función `btree_insert_data` que inserte el elemento `item` dentro del árbol. El árbol pasado como argumento será ordenado: por cada `node` todos los elementos menores serán colocados a la izquierda y los elementos de igual o superior valor serán colocados a la derecha. Para comparar los valores, se pasará una función similar a `strcmp` como argumento.
- El parámetro `root` apunta al nodo raíz del árbol. La primera vez que se llame, deberá apuntar a `NULL`.
- Aquí tienes el prototipo:

```
void btree_insert_data(t_btree **root, void *item, int (*cmpf)(void *, void *));
```

Capítulo IX

Ejercicio 05 : btree_search_item

	Ejercicio: 05
btree_search_item	
Directorio de entrega: *ex*05/	
Archivos a entregar: `btree_search_item.c, ft_btree.h`	
Funciones autorizadas: `Ninguna`	

- Crea una función `btree_search_item` que devuelva el primer elemento relacionado con el dato del puntero dado como referencia. Debes buscar en el árbol recorriéndolo de forma `inordinal`. Si el elemento no se encuentra, la función devolverá `NULL`.
- Aquí tienes el prototipo:

```
void *btree_search_item(t_btree *root, void *data_ref, int (*cmpf)(void *, void *));
```

Capítulo X

Ejercicio 06 : btree_level_count

	Ejercicio: 06
btree_level_count	
Directorio de entrega: $ex06/$	
Archivos a entregar: `btree_level_count.c, ft_btree.h`	
Funciones autorizadas: `Ninguna`	

- Crea una función `btree_level_count` que devuelva la longitud de la rama más larga del árbol pasado como argumento.
- Aquí tienes el prototipo:

```
int btree_level_count(t_btree *root);
```

Capítulo XI

Ejercicio 07 : btree_apply_by_level

	Ejercicio: 07
btree_apply_by_level	
Directorio de entrega: *ex*07/	
Archivos a entregar: `btree_apply_by_level.c, ft_btree.h`	
Funciones autorizadas: `malloc, free`	

- Crea una función `btree_apply_by_level` que ejecute la función, pasada como segundo argumento, a cada nodo del árbol. El árbol debe ser recorrido nivel por nivel. La función dada como argumento recibirá tres argumentos:
  - El primer argumento, de tipo `void *`, será el `item` del nodo actual.
  - El segundo argumento, de tipo `int`, será el nivel en el que nos encontremos: 0 para la raíz, 1 para los hijos, 2 para los nietos, etc.
  - El tercer argumento, de tipo `int`, será `1` si es el primer `node` del nivel, o `0` de otro modo.
- Aquí tienes el prototipo de la función:

```
void btree_apply_by_level(t_btree *root, void (*applyf)(void *item, int current_level, int is_first_elem));
```

Capítulo XII

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.