Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2133 - Estructura de Datos y Algoritmos

# Informe Tarea 0

28 de marzo de 2024
Lucas Fernández
20639112

---

## Motivación

Este problema nos permite aprender diferentes habilidades que son de alta utilidad para desarrollos futuros. Por una parte, se aprende el correcto uso de C, tanto en control de flujo como uso correcto de la memoria, lo que nos entrega la capacidad de desarrollar con un lenguaje de bajo nivel y, así, conseguir programas de mayor velocidad que con lenguajes previamente aprendidos como Python. Por otra parte, nos entrega la oportunidad de conocer diferentes estructuras y algoritmos que se utilizan usualmente en desarrollos, por ejemplo, el uso de arrays, listas ligadas y algoritmos de sorting, esto nos da la capacidad se expande nuestros conocimientos y mejorar nuestro desarrollo y estimaciones de tiempo y comportamiento sobre estos.

## Informe

Para solucionar el problema se utilizaron tres estructuras, Planet, Ship y Order. Planet contiene su id y la cuenta de las órdenes que se han entregado en el planeta. Ship contiene su id y un puntero a la siguiente orden que debe entregar. Order contiene su id, el planeta en el cual debe ser entregado y un puntero a la orden que viene luego de ser esta entregada, generando así una lista ligada.

Los planetas y naves se deben de guardar a lo largo del problema, para esto se utilizó un array de punteros a las diferentes estructuras, para lo cual se reservó memoria para el array y para los punteros a las estructuras. Luego, los punteros a los pedidos se incorporaban inicialmente a la nave, si esta no tenía un pedido, y, a continuación, si la nave ya tenía algún pedido, el puntero al nuevo pedido se incorporaba en el último pedido, de esta forma se consigue una lista ligada con funcionamiento FIFO. Para los punteros de los pedidos también se reserva la memoria correspondiente. Finalmente, luego de resolver el problema, se realizaba la liberación de la memoria, para esto, se recorría el array de los planetas y las naves y se liberaba la memoria de cada uno, para luego, liberar la memoria de los arrays. Previo a liberar la memoria de las naves se liberaba la memoria de los pedidos, para esto se ocupaba una función recursiva la cual, si el pedido tenía un siguiente pedido, se llama la función con el siguiente pedido y, de no tener un siguiente pedido, se liberaba la memoria y realizaba lo mismo hacia 'atrás'.

Para la implementación del comando PEDIDO-CONTAMINADO se recorre la lista ligada de pedidos de la nave guardando el pedido previo al que está en revisión y si se logra encontrar el pedido contaminado, se actualiza el siguiente pedido del pedido previo como el siguiente del ligado, se elimina el pedido contaminado, liberando la memoria y, en el caso de ser el primer pedido, se actualiza el siguiente pedido de la nave. En el caso de no encontrar el pedido se retorna 0 e imprime lo asociado.

Para la implementación del comando COORDINAR-PEDIDOS inicialmente se cuentan las órdenes de cada nave al determinado planeta, esto se realiza recorriendo la lista ligada y contando los órdenes que tienen el ID del planeta. Luego, dependiendo de qué nave tiene más pedidos, se recorren los pedidos de la nave con menos cantidad de pedidos al planeta

determinado. Cuando se identifica una orden que va al determinado planeta se le asigna a la orden previa la siguiente orden y luego se le quita la siguiente orden a la orden identifica para luego ser agregada a la nave con más pedidos. Se cuentan los pedidos transferidos y se retorna esta cuenta para imprimirla como se solicita. Existe el caso donde alguna de las dos naves no tiene pedidos para el planeta solicitado, en este caso se retorna 0.

Para invertir una lista ligada se deben utilizar tres variables, una que contenga la orden actual, otra la siguiente y, la última, la orden anterior. Luego, se guarda la siguiente orden, se toma la orden actual, se le asigna como siguiente la orden previa y luego se iguala la variable de orden anterior a la variable actual y la variable actual a la orden siguiente. De esta forma se recorre la lista original mientras se arma la lista invertida a medida que se recorre.

Por último, para implementar el ordenamiento de los planetas, primero, se realizó una copia de la lista original, para evitar desordenar la lista original. Luego, se iteró con un doble for a través de los índices, partiendo el segundo for una posición delante del primero. Dentro de esta iteración se revisaba si el elemento correspondiente al índice del primer for era menor a los siguientes, en el caso de ser menor se intercambiaba posición con el elemento. Luego, si ambos elementos eran iguales, se revisaba su id y se realizaba el intercambio en el caso de tener id mayor que alguno de los siguientes elementos.

# Conclusión

Concluyendo, el problema planteado nos entregó la oportunidad de comenzar nuestra ruta de aprendizaje de C, lenguaje de gran utilidad. También nos permitió poner en práctica los conocimientos, por una parte, sobre estructuras de listas y listas ligadas y, por otra parte, sobre algoritmos de ordenamiento. Por último, el uso de C en este problema fue de gran utilidad para expandir el conocimiento sobre uso de memoria y como el mal uso de esta puede impactar el rendimiento de los computadores.