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

Informe Tarea 0

27 de marzo de 2024
Manuel Espinoza Quintero
20642598

Motivación

Este programa está diseñado para manejar pedidos de entrega en el contexto de naves espaciales que viajan a diferentes planetas. La lógica central del programa se organiza alrededor de la interacción entre tres estructuras de datos principales: `Order`, `Planet`, y `Ship`, y se controla mediante una serie de comandos que determinan el flujo del programa. Sin embargo, este sistema tiene aplicaciones prácticas más allá del contexto espacial de la tarea, especialmente en logística y distribución. Este tipo de modelado es crucial para optimizar rutas de entrega, manejar inventarios dinámicos y asegurar la eficiencia operativa. La habilidad para ajustar pedidos, coordinar entregas y manejar recursos que cambian refleja los desafíos reales de algunas industrias reales.

Informe

Estructuras Principales

`Order` (Definida en `order.h`)

La estructura `Order` representa un pedido individual. Cada pedido tiene un identificador único (`id`), un identificador del planeta destino (`planet_id`), y un puntero al siguiente pedido en la lista (`next`). Esta es una estructura de lista ligada clásica, donde cada pedido sabe cuál es el siguiente pedido en la secuencia, permitiendo una gestión dinámica de los pedidos.

`Planet` (Definida en `planet.h`)

La estructura `Planet` representa un planeta, con un identificador único (`id`) y un contador de entregas exitosas (`succesfull_deliveries`). Esta estructura se utiliza para rastrear el número de pedidos entregados con éxito a cada planeta.

`Ship` (Definida en `ship.h`)

La estructura `Ship` representa una nave espacial, con un identificador único (`id`) y un puntero al primer pedido en su lista de pedidos (`orders_head`). Esta estructura utiliza una lista ligada de pedidos para gestionar dinámicamente los pedidos asignados a la nave.

Implementación de comandos

Comando `PEDIDO-CONTAMINADO`

El comando se encarga de eliminar los pedidos problemáticos y se implementa siguiendo estos pasos:

1. **Identificar el Pedido:** Se obtiene la información del archivo de entrada sobre cuál nave y qué pedido específico está contaminado.

2. **Buscar el Pedido en la Nave:** La nave especificada es buscada en su lista de pedidos para encontrar el pedido contaminado. Para esto se utiliza un puntero doble (`Order **current`) , permitiendo modificar el puntero al pedido actual (`*current`).
3. **Eliminar el Pedido:** Una vez encontrado, el pedido contaminado es eliminado de la lista, manteniendo intacta la secuencia de los pedidos restantes, esto implica cambiar el puntero del pedido anterior para que apunte al siguiente pedido del contaminado, sacándolo de la cadena.
4. **Confirmación:** Se registra en el archivo de salida la eliminación del pedido contaminado.

Comando `COORDINAR-PEDIDOS`

El comando intenta equilibrar los pedidos entre naves para un planeta específico, y se lleva a cabo de la siguiente manera:

1. **Recopilar Información:** Se identifican las dos naves involucradas y el planeta objetivo para la coordinación de pedidos.
2. **Contar Pedidos:** Se cuenta cuántos pedidos para el planeta especificado tiene cada nave, recorriendo la lista ligada y revisando `planetID`.
3. **Determinar Acción:** Se decide cuál nave será la dadora y cuál la receptora de los pedidos adicionales, basándose en quién tiene más pedidos para el planeta.
4. **Transferir Pedidos:** Se transfieren pedidos del planeta especificado de la nave con más pedidos a la nave con menos, para esto se elimina de la lista ligada de la nave fuente ajustando los punteros correspondientes, y se añade al final de la lista de la nave destino. Esto requiere encontrar el último pedido de la nave destino y ajustar su puntero `next` para que apunte al pedido transferido. Luego, el puntero `next` del pedido transferido se establece en `NULL`, marcándolo como el nuevo final de la lista.
5. **Registro y Confirmación:** Se documenta en el archivo de salida la cantidad de pedidos transferidos y la distribución final entre las dos naves.

Invertir una lista ligada:

Para invertir una lista ligada, se utilizan tres punteros: `prev` (inicialmente `NULL`), `current` (apunta al primer nodo), y `next` (guarda el siguiente nodo temporalmente). Se recorre la lista, haciendo que cada nodo apunte al anterior. Se actualizan `prev` y `current` para avanzar, con `prev` tomando el lugar de `current`, y `current` el de `next`. Este proceso termina cuando `current` es `NULL`, y `prev` señala al nuevo primer nodo de la lista invertida.

Ordenamiento de planetas

El ordenamiento de los planetas por la cantidad de pedidos se implementa mediante la función `qsort` [1] de C, que aplica QuickSort, la cual requiere un arreglo para ordenar su tamaño, el tamaño de cada elemento y una función de comparación. La función de comparación, `comparePlanets`, compara dos planetas primero por su cantidad de entregas exitosas y, en caso de empate, por su identificador. Si un planeta tiene más entregas exitosas que otro, se considera mayor. En caso de igual número de entregas, el de menor identificador se considera mayor para mantener un criterio de orden consistente. Esto asegura que el arreglo de planetas se ordene de manera descendente por entregas exitosas y, en caso de empate, de manera ascendente por su identificador, tal como lo indica el enunciado.

Conclusión

El código diseñado para manejar pedidos, naves y planetas refleja directamente las necesidades prácticas del enunciado. Con estructuras simples como listas ligadas y arrays, permite añadir, quitar y reorganizar pedidos fácilmente, además de coordinar entre naves y seguir entregas a planetas. Esto hace que el sistema sea flexible y directo para gestionar la logística de entregas, asegurando que los comandos tengan una fácil implementación.

Referencias

[1] GeeksforGeeks, *Comparator function of qsort() in C*, [online] Available: `https://www.geeksforgeeks.org/comparator-function-of-qsort-in-c/`.