Estructuras de datos del MIPS R2000

Fausto Leoni, Sebastian Braida, Lucas Ojeda, Juan Cruz D'Amor, Lautaro Lazzaroni

20 de agosto de 2021

1. Estructura de una lista doblemente enlazada

Lo primero que hicimos fue determinar con palabras como funciona la actividad propuesta. Con ayuda de las ilustraciones en el TP fuimos determinando cada campo de cada tipo de nodo (entendemos campo por cada word, cada nodo tiene un total de 4 words = 16 bytes = 4 campos).

De esta forma, los nodos de categoria tienen: dos campos asignados para el proximo y el anterior nodo (primer y ultimo campo respectivamente), en el segundo campo tenemos la direccion de memoria de el primer objecto de la categoria, y por ultimo tenemos en el tercer campo la direccion de memoria del dato, que en este caso es el nombre de la categoria.

Despues tenemos el objeto, el cual funciona de una froma similar al de categoria en el primer y ultimo campo, a su vez el tercer campo funciona de una forma similar, teniendo la direccion de memoria del nodo de tipo dato que indica el nombre del objecto. A diferencia del segundo campo de la categoria, en el objecto vamos a ver el ID que es una clave unica que identifica a un objecto dentro de una categoria.

Por ultimo, tenemos los nodos de tipo Dato, estos utilizan las 4 words para almacenar una cadena de caracteres.

2. Menu

El menu en la consola permite al usuario interactuar con el programa. Sus opciones son:

1. Crear una nueva categoria vacia.
2. Seleccionar la siguiente categoria.
3. Seleccionar la anterior categoria.
4. Listar todas las categorias.
5. Borrar la categoria seleccionada.
6. Anexar un objeto a la categoria seleccionada.
7. Borrar un objeto de la categoria seleccionada.
8. Listar los objetos de la catogoria seleccionada.
9. Salir del programa.

El usuario debe ingresar el numero de la opcion que desea para seleccionarla y luego, en caso de que la opcion lo requiera, se le pedira mas informacion. Las opciones que requieren mas informacion son:

Opcion 1 Requiere el nombre de la categoria.

Opcion 6 Requiere el nombre del objeto.

3. Funciones generales

Debido a las similitudes entre todos los tipos de nodos pudimos crear dos funicones utilizadas en varias partes del codigo que son generales, indistintamente de si son llamadas para una categoria o un objeto. Estas funciones son: addNode y delNode

3.1. addNode

Es una funcion que tras recibir el diriccion del primer nodo de la lista, ya sea de objectos o de categorias, y la direccion donde se encuentra ese primer nodo como segundo parametro, esto va a devolver la direccion del nuevo nodo creado.

Internamente la funcion se encarga de reservar memoria para el nuevo nodo, luego se encarga de asignar la direccion de memoria del nodo de dato al tercer campo del nodo. En caso de que sea el primer nodo se asignara el primer y ultimo campo con la direccion de memoria del propoio nodo, ya que es una lista circular, por otro lado tenemos el caso de que no sea el primer nodo, el cual se enlazara de forma que el proximo del primer nodo ahora apunte al nuevo nodo y que el anterior del viejo ultimo nodo ahora apunte al nuevo nodo. A su vez el nuevo nodo estara doblemente enlazados al primer nodo y al viejo ultimo nodo.

De esta forma tenemos tres de los cutro campos completos, a excepcion del segundo campo, el cual sera completado dependiendo quien llame esta funcion.

3.2. delNode

Esta funcion recibe la direccion del nodo que se desea eliminar y no tiene valor de retorno. delNode se encarga de vincular el nodo anterior y el siguiente (respecto al nodo que se desea borrar) entre si. Luego llama a sfree para liberar la memoria utilizada tanto por el nodo como por el dato de ese nodo.

4. Funciones no generales

4.1. newCategory

newCategory se encargara de llamar al addNode creando la nueva categoria, luego si la nueva categoria es la primera en ser creaada asignara la direccion de la nueva categoria a wclist (working category).

4.2. prevCategory y nextCategory

Este set de funciones se encarga de modificar wclist para que apunte a la anterior o a la siguiente categoria respectivamente.

4.3. displayCategories

displayCategories itera a travez de las categorias existentes mediante el uso de la lista doblemente enlazada. En el caso de que el nodo el cual esta cargado sea tambien el nodo en el wclist, esto hara que se muestre en la consola un '*' a forma de identificar la categoria seleccionada.

4.4. delCategory

delCategory se encarga de evaluar si hay un solo nodo, en ese caso asigna cclist a null, en caso contrario analiza si la categoria que se desea eliminar es la primera de la lista, de ser asi asigna cclist al segundo nodo de la lista ya que ahora este pasaria a ser el primer nodo. En culaquier otro caso cclist se mantendra con su valor actual ya que

no se elimina la primer categoria de la lista.

Si la categoria a borrar tiene objetos asociados, itera a travez de ellos eliminando uno por uno.

Por ultimo asigna la wclist al primer nodo de la lista y finalmente llama a delNode.

4.5. newObject

newObject se encarga de llamar a addNode. Luego de esto verifica si el nodo que se creo fue el primer objeto de la categoria. En ese caso asignara un 1 como el id del objeto creado, si no asignara el id como el id del objeto anterior incrementado en 1.

4.6. delObject

delObject se encargara de fijarse si el nodo que se desea eliminar es el primero, de ser cierto verificara si este objeto es el unico de la categoria. En ese caso el puntero del segundo campo de la categoria seleccionada se asignara a null, indicando que la categoria ya no tiene mas objetos. En el caso de que sea el primero pero no sea el unico objeto de la categoria, asignara el puntero mencionado al segundo objeto de la categoria. En ambos casos, se procede a eliminar el objeto utilizando delNode.

En el caso de que no sea el primer objeto de la categoria, la funcion itera a travez de todos los objetos hasta encontrar el objeto con el id deseado y lo borra mediante delNode.