Instituto Tecnológico de Costa Rica

Sede Interuniversitaria

Ingeniería en Computación

Estructuras de Datos - I Proyecto Programado

Profesora: Samanta Ramijan Carmiol

Estudiantes: Natán Fernández de Castro - 2017105774
Pablo Venegas Sánchez - 2018083497

Documentación Externa

TEC-To-Do Organizador de Tareas en Lenguaje C

II Semestre 2018

1. Tabla de Contenidos

2. Descripción del problema

Cuando de organización se trata, es conveniente hacer uso de una herramienta que facilite una estructuración de la tareas, ya sea del día a día o esporádico. En sí el problema es mediante un programa, un usuario pueda tener categorías en las cuales ingrese tareas. Específicamente, mediante el uso de conceptos de Estructuras de Datos, tales como: manejo de punteros, listas enlazadas simples o dobles y manejo de memoria; crear nodos mediante terminal que permitan al usuario visualizar sus tareas pendientes, ordenadas por colores. El programa debe permitir, a través del uso de comandos, agregar, eliminar o mostrar categorías y tareas. Cuando logre completar una tarea puede marcar esta como hecha.

Como parte adicional, el programa debe hacer uso de archivos .txt para asegurar el estado de las categorías y tareas, cuando el usuario cierre o abra el programa se mantendrá la configuración. Además, implementar el uso de makefile para compilar y ejecutar en terminal el archivo de programa.

3. Diseño del programa

3.1 Decisiones de diseño

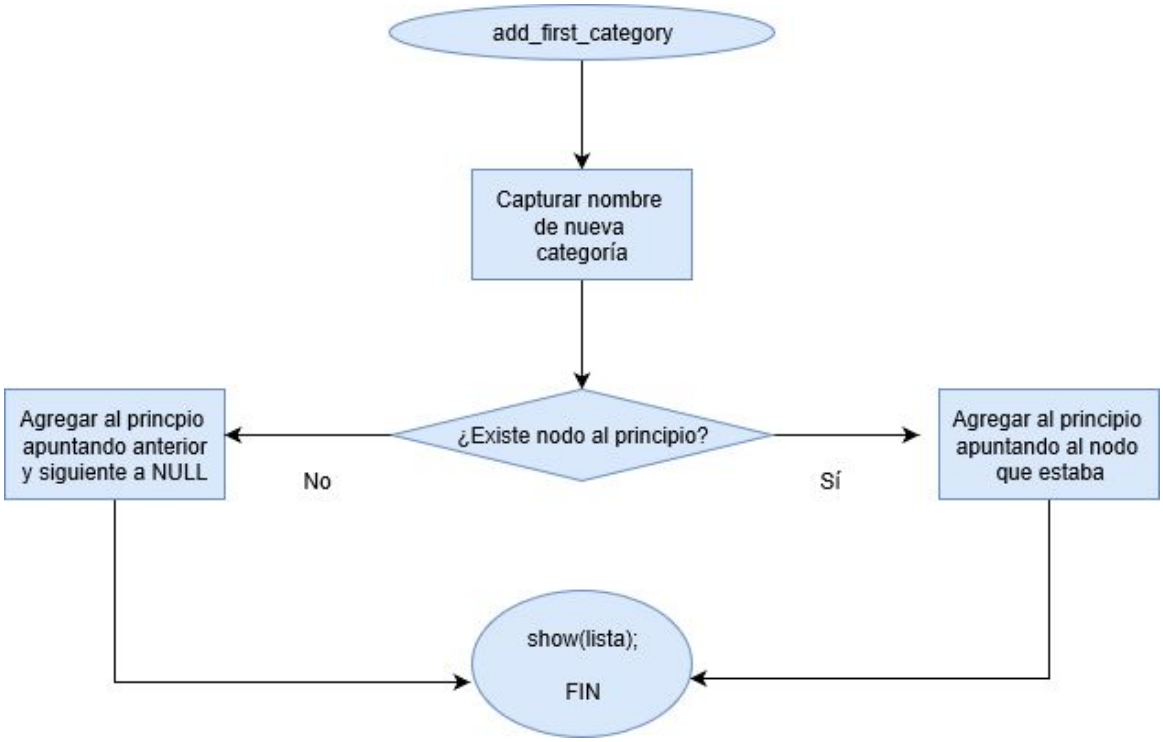
Con respecto a la solución del problema, se implementa una lista doble enlazada para manejar las categorías, cada nodo de la categoría contiene tres punteros: siguiente (Apunta al nodo siguiente), anterior (Apunta al nodo anterior), primer_tarea (Apunta a la primer tarea que contiene ese categoría) y ultima_tarea (Apunta a la última tarea que contiene la categoría). Estos dos últimos punteros son necesarios para la utilización de una lista simple, contenida dentro de cada categoría. En sí la estructura nodo contiene: cuatro punteros(mencionados anteriormente), nTareas (Entero que almacena el número de tareas en esa categoría) y un char categoría(Almacena el nombre de la categoría).

Dentro de cada categoría dos punteros, mencionados anteriormente, apuntan a una lista simple que contendrá las tareas. Esta estructura de lista enlazada simple contiene: Un puntero siguiente (apunta al siguiente nodo en la lista), char tarea (Almacena el nombre de la tarea) y un char estado(Que constituye en un símbolo, ya sea un cuadro vacío o con una marca, para mostrar si la tarea está pendiente de hacer o no. Para la solución del problema se haría uso de un manejo apropiado de estas listas enlazadas. Aplicando conceptos de recorrido, eliminación y manipulación de las listas enlazadas.

3.2 Algoritmos

- add_first_category: Se captura el nombre que el usuario quiere crear la categoría, analiza si tiene un nodo al principio ya creado o si no hay ninguno creado al momento. De ser positivo agrega la nueva categoría, se apunta a la categoría que estaba anteriormente pasando esta a ser segunda.
- add_last_category: Captura el nombre que usuario quiere crear la categoría, muy parecido al procedimiento de agregarlo al principio. Solo que a la hora de insertar la categoría se analiza si hay algún nodo al final y se apunta con el anterior en caso de que exista un nodo.
- delete_category: Captura el nombre de la categoría a eliminar, recorre la lista enlazada doble, si lo encuentra lo procede a eliminar revisando los nodos que tenga alrededor y dependiendo de eso se redirige los punteros
- add_to_do: Captura la categoría y la tarea que desea que desea agregar. Recorre primero la lista enlazada doble. Una vez que encuentra la categoría recorre la lista simple hasta que apunte a null y agrega la tarea al final. En el caso que no tuviera tareas esa lista, la agrega al principio.
- mark_to_done: Para esta función se aplican casi los mismos pasos que para agregar tarea, a diferencia que, luego de recorrer la lista doble y la lista simple, tiene que encontrar la tarea a la que hay que marcar como hecha. La solución al problema que se aplicó fue: cuando una tarea esté pendiente, tendrá el caracter de cuadro vacío; cuando se quiera marcar como hecha tendrá el caracter de cuadro con una equis dentro.
- delete_to_do: La función inicia igual, captura la categoría a buscar; luego la tarea. Una vez que encuentra la tarea procede a liberar el espacio en memoria de ese nodo mediante el la función free(); . Tomando en cuenta los nodos que tenga la lista simple.
- show current category: Función que muestra la categoría actual mediante impresión del nodo actual de la lista doble. Además se recorre con un for() la lista simple para imprimir las tareas de esa categoría.
- show_next_node: Mismo proceso, se imprime el nodo, solo que esta vez el siguiente del actual, de igual manera, con las tareas que posea para mostrarlas en pantalla.
- show_previous_category: Muestra mediante impresión el nodo anterior al que apunta el nodo actual.

3.3 Diagrama



4. Análisis de resultados

Los resultados fueron exitosos en un 90%, esto debido a que no se pudo implementar de la mejor manera la carga de archivo .txt. Por lo demás se completó de la mejor manera.

Con respecto al grupo de trabajo, las tareas se dividieron manteniendo siempre orden y buena comunicación.

5. Manual del usuario

Mediante uso de un makefile, dentro de la ruta del archivo enlazada.c, se escribe en terminal:

```
>>> make
>>> ./enlazada
```

Esto ejecuta el archivo en donde ya se puede usar el programa a través de los siguientes comandos:

add_first_category: Agregar una nueva categoría al inicio de la lista
add_last_category: Agregar una nueva categoría al final de la lista
delete_category: Eliminar una categoría
add_to_do: Agregar una tarea a una categoría específica
mark_to_do_done: Marcar una tarea como realizada
delete_to_do: Eliminar una tarea
show_current_category: Mostrar la categoría actual

show_next_node: Mostrar la siguiente categoría
show_previous_category: Mostrar la categoría anterior
help: Mostrar lista de comandos disponibles
about: Mostrar información del programa, curso y desarrolladores

6. Lecciones aprendidas

- Uso de un sistema operativo para grupos de trabajo.
- Manejo de listas enlazadas.
- Creación de archivo make y su utilización en el programa.
- Ampliación en el conocimiento de uso de C.
- Comportamiento de C con la memoria. Esto debido a problemas con impresión de basura (caracteres aleatorios) en memoria.