Taller de Lenguajes I – 2023
Programador Universitario / Licenciatura en Informática / Ingeniería en Informática
Trabajo Práctico N° 4

OBJETIVOS

- Manejar arreglos dinámicos.
- Manejo de Listas enlazadas.
- Llevar a cabo la codificación de programas utilizando estructuras como un tipo de datos.
- Manejo de conflictos al realizar la operación Merge en Git.

Ejercicios:

1) Si todavía no creó el repositorio, copie el siguiente enlace en su navegador: https://classroom.github.com/a/StFLiI_4 esto creará el repositorio para poder subir el **Trabajo Práctico Nro. 4**, realice los pasos ya aprendidos para *clonar* el repositorio en su máquina y poder comenzar a trabajar de forma *local*.

   Nota. No se olvide de incluir el archivo .gitignore en la raíz del repositorio para excluir los archivos .exe, .obj y .tds del mismo.

2) Considere la siguiente situación: En la misma distribuidora del práctico anterior ahora surgió la necesidad de llevar un control de las tareas realizadas por sus empleados. Usted forma parte del equipo de programación que se encargará de hacer el módulo en cuestión que a partir de ahora se llamará módulo ToDo:

   Tareas

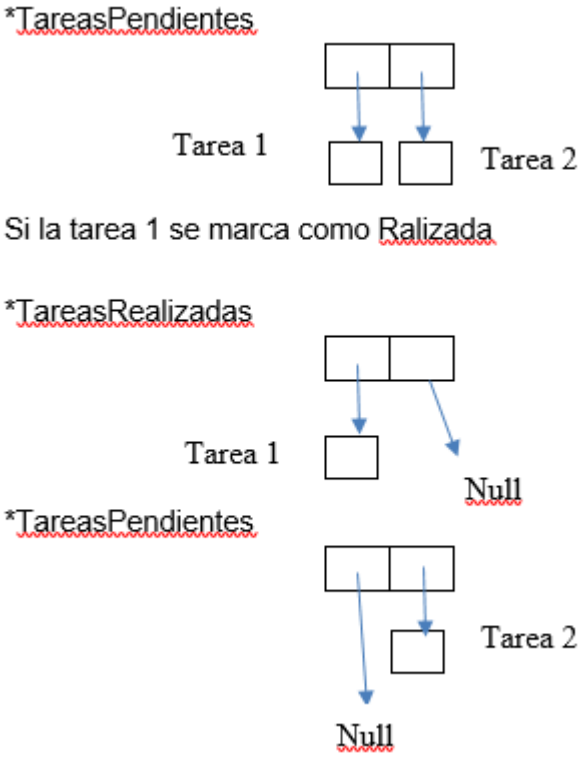
   Cada empleado tiene un listado de tareas a realizar.

   Las estructuras de datos necesarias son las siguientes:

```
struct Tarea {
            int TareaID;     //Numerado en ciclo iterativo
            char *Descripcion;
            int Duracion; // entre 10 – 100
};
```

   1. Desarrollar una interfaz por consola donde se solicite al usuario (es decir el empleado) cuantas tareas debe cargar.
   2. Tendrá que generar un arreglo de doble punteros dinámicamente del tipo Tarea con la cantidad de tareas solicitadas en el punto anterior.Recuerde inicializar los arreglos apuntando a NULL.
   3. Desarrolle una interfaz de consola para cargar las tareas, ingresando su descripción y duración. Recuerde utilizar reserva de memoria dinámica para la carga de la descripción.

4. Una vez cargada todas las tareas. Irá listando de a una las tareas y preguntando si se realizó o no la misma. Si la respuesta es Si tiene que "mover" dicha tarea a otro arreglo denominado tareas realizadas que tendrá el mismo tamaño que el anterior. Una vez movida la tarea ese casillero del vector tiene que apuntar a null. Como se muestra en la gráfica a continuación.



5. Mostrar por pantalla todas las tareas realizadas y luego listar las tareas pendientes.
6. Cree un nuevo branch llamado *busca-tarea* e implemente una función de búsqueda de tarea por nro. de id de nombre *BuscarTarea*. La misma devuelve la tarea solicitada.
7. Vuelva al branch *main* e implemente también una nueva versión de la función *BuscarTarea* en donde la misma sea por palabra clave en vez de por Id. (uno le manda una palabra y te tiene que devolver la primera tarea que contenga dicha palabra).
   Nota: Para realizar este punto, investigue el uso de la función **strstr**.
8. Realizar el merge correspondiente y resuelva el conflicto producido. Finalmente modifique los nombres de las funciones a *BuscaTareaPorId* y otra *BuscaTareaPorPalabra*.
9. Agregue una interfaz de usuario al programa principal que permita consultar tareas por id y palabra clave y mostrarlas por pantalla.

3) Cree un nuevo branch llamado *lista-enlazada* donde tendrá que reemplazar la implementación que realizó con listas enlazadas, para ello en vez de tener 2 arreglos de tareas ahora tendrá 2 listas enlazadas (una para las tareas pendientes y otra para las tareas realizadas) y cada vez que se marque como realizada tendrá que mover la tarea de la lista de tareas pendientes a la lista de tareas realizada

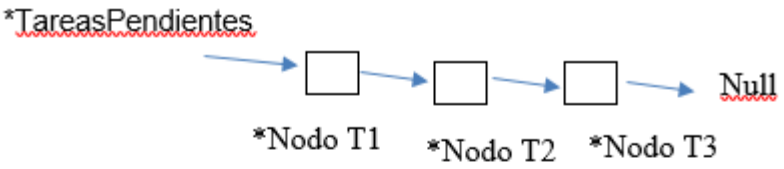
Tareas

Cada empleado tiene un listado de tareas a realizar.

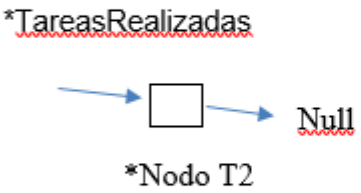
Las estructuras de datos necesarias son las siguientes:

```
struct Tarea{
        int TareaID;    //Numerado en ciclo iterativo
        char *Descripcion;  //
        int Duracion; // entre 10 – 100
};
struct Nodo{
        Tarea T;
        Nodo *Siguiente;
};
```

Observe la gráfica:



Si la tarea 2 se marca como Realizada



1. Reimplemente los puntos 1 a 5 del punto anterior utilizando las listas enlazadas. Modifique la interfaz de carga de tareas para que en vez de solicitar la cantidad de tareas al comienzo, al cabo de cada tarea pregunte al usuario si desea ingresar una nueva tarea o finalizar la carga.

2. Implemente también las funciones de búsquedas para que devuelvan el nodo correspondiente a la tarea solicitada Cree también la interfaz de usuario de consulta de tareas.