Haga el código modular, reutilizable, simple. Todos estos criterios se consideran en la rúbrica de evaluación. Esta reúbrica se encuentra al final de este documento

Operaciones

Este proyecto debe cumplir todas las especificaciones del Taller1: enums, recursión y structs. Tenga en cuenta el uso de adecuado de la recursión, la recursión no se debe usar por ejemplo para hacer validaciones o para llamar menus.

Las nuevas operaciones que debe incluir su centro comercial son:

- Información extra en el struct: en total el struct que representa cada local del centro comercial debe tener al menos seis campos: de los cuáles al menos 1 debe ser un enum y otro debe ser numérico.
- Manejo de archivos binarios: debe poder cargar y guardar la información del centro comercial (sus locales) en un archivo binario considerando que esta información se puede actualizar durante la ejecución de su programa. Por ejemplo se pueden agregar locales, cambiar el estado de los locales, etc.
- Ordenamiento iterativo de toda la matriz. Cree dos funcionalidades para ordenar la matriz que representa el centro comercial. Una de sus operaciones debe usar el algoritmo de selección y otra debe usar el algoritmo de inserción.
- Ordenamiento recursivo. Cree funcionalidades para ordenar usando merge sort y quick sort los pisos de la matriz. Una funcionalidad para cada tipo de algoritmo. Por ejemplo, podría ordenar el piso por número de empleados, por valor de las ventas semanales, por cantidad de productos en el inventario, etc.
- Manejo de excepciones: Incluya en su programa el manejo de excepciones funcionales para garantizar que los datos se encuentren en los rangos correctos, que se pueda leer y guardar los archivo correctas y demás partes de su programa donde puedan ser necesarias.

Deseable [incorpore funcionalidades que usen la estructura pila y la estructura cola]. Según la sustentación que haga de las mismas le daré entre 0 y 2 décimas adicionales en la nota final del parcial]

Mínimos esperados (no cuentan en la nota, pero sin esto no califico el parcial):

En la elaboración de su programa debe considerar:

- Uso continuo de git para mantener el histórico de avance de su proyecto con comentarios claros sobre los cambios de su programa.
- **Informe de mejoras:** documento en el que explique que ajustes hizo de su código anterior para agregar las nuevas funcionalidades o para mejorar el código existente. Tenga en cuenta la retroalimentación que recibió cuando sustentó el parcial 2.
- Tener un menú usando do while y switch case para acceder a las diferentes opciones del programa.
- Tener un makefile para compilar el programa.
- Cumplir con el estándar de codificación lowerCamelCase, en el que las operaciones inician con un verbo en infinitivo. Las palabras compuestas inician en minúsculas y la segunda palabra tiene la primera letra en mayúsculas Ejm: *llenarCentroComercial.* Las variables tienen nombres que semánticamente se relacionan con la función que cumplen. El cumplimiento de este estándar es obligatorio.
- Buena indentación y organización del código. Sin errores graves o warnings luego de revisarlo con el CPP check.
- Código documentado para hacerlo claro para cualquier lector.
- Buenas prácticas de programación: buen nombramiento, no números mágicos, uso de constantes.

Entregables:

Suba a su repositorio de github en una carpeta llamada ProyectoFinal:

- el código fuente de su programa (archivos .c, .h y makefile)
- Nuevo documento en el que explique las funcionalidades que implementó en su programa y para qué sirven, considerando el siguiente ejemplo:
    - Operación *calcularVentasXTipoLocal*.
        - *Entradas*: Recibe el tipo de local para el que se hará la operación.
        - *Salidas*: Valor de la suma de las ventas totales acumuladas de los locales ocupados en el centro comercial para el tipo recibido.
        - *Conceptos usados:* Enums. El tipo de local es un valor del enum y paso de parámetros por referencia pues recibe el apuntador a la matriz del centro comercial.
- Puede subir la información a su repositorio máximo hasta las 11:55 pm del 23 de mayo del 2020.
- Tenga en cuenta que programas que no compilen no serán revisados.

Rúbrica de evaluación

La nota del parcial será dada al finalizar la sustentación del parcial según los criterios que se describen a continuación y su capacidad para sustentar su trabajo.

	5	4	3	2	1	0
Funcionalidad (60%)	Excedió las expectativas	Cumplió con todos los requisitos.	Fueron desarrollados mínimo el 75% de los requisitos	Fueron desarrollados mínimo el 50% de los requisitos	Fueron desarrollados mínimo el 25% de los requisitos	Fueron desarrollados menos del 25% de los requisitos
Estilo de codificación (15%)	El código se encuentra correctamente indentado, los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. El código tiene documentación interna para facilitar la revisión.	La mayoría del código se encuentra correctamente indentado- La mayoría de los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. La mayoría del código tiene documentación interna para facilitar la revisión	Falta una de las cosas del estilo de codificación o alguna se cumple con mala calidad	Faltan dos de las cosas del estilo de codificación o se cumplen con mala calidad:	Faltan tres de las cosas del estilo de codificación o se cumplen con mala calidad:	No cumple con el estándar de nombramiento No se encuentra correctamente indentado No está divido adecuadamente
Mejores prácticas (25%)	El código muestra mejores prácticas de desarrollo siempre. Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.	El código muestra mejores prácticas de desarrollo en la mayoría de los casos, pero falta mejorar algunos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones	El código muestra buenas prácticas de desarrollo pero falta mejorar dos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones	El código aplica pocas buenas prácticas de desarrollo. Falta mejorar tres de los siguientes aspectos: Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.	El código aplica muy pocas buenas prácticas de desarrollo. Falta mejorar cuatro de los siguientes aspectos: Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.	El código no considera buenas prácticas de desarrollo. Falta mejorar cinco o más de los siguientes aspectos: Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.

Rubrica de propiedad intelectual

	1	0.8	0.6	0.4	0.2	0
Sustentación	Es evidente que el estudiante	La sustentación es buena pero se evidenció	La sustentación es aceptable se evidencia que	La sustentación es regular se evidenció	El estudiante demuestra que entiende partes	Se evidencia que el estudiante

	entiende el código que desarrolló lo explica con claridad y responde correctamente a las preguntas.	inseguridad del estudiante para explicar algunas partes del trabajo desarrollado o para responder algunas preguntas.	el estudiante desarrolló el código pero le cuesta trabajo explicar aspectos del código.	inseguridad del estudiante para explicar gran parte del trabajo desarrollado o para responder muchas de las preguntas. Parece que el código no hubiera sido desarrollado por el estudiante.	del código, pero no tiene claro cómo se relacionan con la funcionalidad solicitada.	no entiende el código desarrollado, no es capaz de responder a las preguntas formuladas de manera correcta.