Parcial 2
Técnicas y prácticas de programación
Abril 12 del 2020

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

Operaciones

Descargue el código que se encuentra disponible en el repositorio de ejercicios en la carpeta **CentroComercial.** Use estos archivos y el struct que ya está definido para construir un programa que sirva para:

1) Solicitar el número de pisos del centro comercial (filas) y el número de locales disponibles por piso (columnas) para el centro comercial.
2) Ingresar un nuevo local comercial. Será guardado en el piso y el local que diga el usuario siempre y cuando esta ubicación se encuentre libre. Si el local no está libre solicítele al usuario otra ubicación hasta que encuentra una ubicación disponible. A cada local su programa le deberá calcular automáticamente un identificador único que quedará guardado en el campo id. Cuando cree un local debe solicitar el resto de información que requiera para llenar los datos.
3) Implementar dos operaciones en las que consulte información del centro comercial (las que quiera)
4) Implementar una operación para modificar ya sea la información de uno o de más locales (lo que prefiera)
5) Implementar una operación para eliminar ya sea uno o más locales (lo que prefiera). Eliminar en el contexto del centro comercial significa que el local pasa de estado ocupado a libre.

Nota: tenga en cuenta que en sus operaciones debe incorporar el uso de enums, recursión, paso de parámetros por referencia y memoria dinámica (puede usarlos como quiera y las veces que quiera pero debe usar al menos una vez cada concepto en todo su programa).

Sorpréndeme [incorpore funcionalidades extras a las operaciones que solicito. 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):

Su programa debe:

- 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.

Entregables:

Suba a su repositorio de github en una carpeta llamada Parcial2:

- el código fuente de su programa (archivos .c, .h y makefile)
- un 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 12 de abril del 2020.

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	La mayoría del código se encuentra correctamente indentado- La mayoría de los nombres de	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

	cumplen con el estándar de nombramiento. El código tiene documentación interna para facilitar la revisión.	los atributos y las funciones cumplen con el estándar de nombramiento. La mayoría del código tiene documentación interna para facilitar la revisión				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 entiende el código que desarrolló lo explica con claridad y responde correctamente a las preguntas.	La sustentación es buena pero se evidenció inseguridad del estudiante para explicar algunas partes del trabajo desarrollado o para responder algunas preguntas.	La sustentación es aceptable se evidencia que el estudiante desarrolló el código pero le cuesta trabajo explicar aspectos del código.	La sustentación es regular se evidenció 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.	El estudiante demuestra que entiende partes del código, pero no tiene claro cómo se relacionan con la funcionalidad solicitada.	Se evidencia que el estudiante no entiende el código desarrollado, no es capaz de responder a las preguntas formuladas de manera correcta.