Nombre: Gerardo Galan Garzafox Fecha 2021-03-16

Programa: lab5 Programa# 5

Instructor: Adriana Gonzalez Language: C++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| completo | Todo lo que se diseñó se codificó | OK |  |  |  |  |
| Incluye | Se tienen todas las librerías que se ocupan y ni una de más | OK |  |  |  |  |
| Inicialización | Verificar variables y parámetros de inicialización  -cuando inicia el programa  -al inicio de cada ciclo  -en las entradas de clases/funciones/procedimientos | X | X | OK |  |  |
| llamadas | Revisar formato  -pointers  -parametros  - & | OK |  |  |  |  |
| nombres | Que sean  -consistentes  -representativos de su contenido  -dentro de su localidad  -no hay acentos ni letras ñ  -referencia con “.” para clases  -clases tienen PascalCase (excepto main en c++)  -variables y funciones tienen camelCase  -constantes tienen mayúsculas VARIABLE\_CONSTANTE | OK |  |  |  |  |
| pointers | Asegurarse de que  -sean inicializados como NULL  -que se borren después de crearse  -que se borren cuando ya no se ocupen | OK |  |  |  |  |
| Formato de Output | -asegurarse de poner los espacios de indentación en el output sean correctos  -asegurarse de tener saltos de línea donde se requieran | OK |  |  |  |  |
| (),{} pares | Asegurarse de que cada () y {} tengan su par | OK |  |  |  |  |
| Operadores logicos | Asegurarse que las estructuras de control (if,else,switch)y ciclos(while, for, do..while) tengan uso apropiado de ==, =, ||, <, >, >=, <=, &&, !, != | X | OK |  |  |  |
| Verificación linea-por-linea | -puntuación correcta  -sintaxis de puntuación | X | OK |  |  |  |
| Estandares | Asegurarse de que el código sigua estándares | OK |  |  |  |  |
| Archivos | Asegurarse de que cada archivo  -se declare debidamente  -se abre debidamente  -se cierre debidamente | OK |  |  |  |  |
| logica | -Siempre tener un return en las funciones y métodos, fuera de los condicionales incluyendo voids.  -Asegurarse de que las comparaciones tengan el mismo tipo de dato | OK |  |  |  |  |
| OOP | -Utilizar getter y setters  -todos las llamadas a métodos lleven ()  -asegurarse de que cada clase hijo pueda reemplazar a su clase padre | OK |  |  |  |  |
| estructura | -que ninguna línea sobrepase los 80 caracteres de longitud  -hay un espacio en blanco separando los métodos y funciones.  -los elementos de un estatuto deben de estar separados por un espacio en blanco  -una literal numérica con decimales debe de tener un dígito antes y otro después del punto  -las expresiones solo deben de usar los paréntesis requeridos  El código solo debe de tener variables que se utilizan  -que los if’s multilínea tengan 2 espacios de indentación  -que lo que vaya dentro de funciones, condicionales y ciclos tengan 4 espacios/ 1 tab de indentación. | X | OK |  |  |  |
| Comentarios | -Hay comentarios descriptivos, no obvios, en secciones donde se ocupa una mayor aclaración del objetivo  -No existen líneas de código sin uso que estén como comentario.  -Los comentarios de funciones/métodos deben contener una breve descripción del propósito de:  la función, sus parámetros y el valor de retorno.  -Las primeras líneas de los archivos de código deben incluir al menos el siguiente bloque de comentarios  --Descripción breve del programa/archivo.  --Una línea con el autor: nombre y matrícula.  --Una línea con la fecha de creación/modificación.  -Las funciones/métodos siempre deben incluir comentarios antes de su definición, excepto los constructores, getters y setters. | OK |  |  |  |  |
| Contabilización | Todas las clases tienen sus debidos tokens de contabilización | OK |  |  |  |  |
| Reuso | Se reutilizaron clases ya existentes en caso de que sean de utilidad para el proyecto | OK |  |  |  |  |