PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
7ma práctica (tipo b)
Primer Semestre 2021

Indicaciones Generales:

- Duración: 110 minutos.

Obligatoriamente los alumnos deberán mantener en todo momento el AUDIO Y VIDEO de sus computadores abierto de modo que puedan recibir los comunicados que se hagan durante el examen y la revisión de los trabajos que estén desarrollando. De tener algún problema deberán hacérselo saber de inmediato al profesor de su horario por correo. De no hacerlo, no se aceptarán reclamos alegando que no oyeron las indicaciones.

- No se pueden emplear estructuras ni **variables globales** (con excepción de los elementos de iostream, iomanip y fstream). Tampoco se podrán emplear las funciones malloc, realloc, strdup o strtok, igualmente no se puede emplear cualquier función contenida en las bibliotecas stdio.h, cstdio o similares y que puedan estar también definidas en otras bibliotecas.
- Deberá modular correctamente el proyecto en archivos independientes. LAS SOLUCIONES DEBERÁN DESARROLLARSE BAJO UN ESTRICTO DISEÑO DESCENDENTE. Cada función NO debe sobrepasar las 20 líneas de código aproximadamente. El archivo **main.cpp** solo podrá contener la función **main** de cada proyecto y el **código contenido en él solo podrá estar conformado por tareas implementadas como funciones**. En cada archivo que implemente en los proyectos (.h y .cpp) deberá colocar un comentario en el que coloque claramente su nombre y código, de no hacerlo se le **descontará 0.5 puntos por archivo**.
- NO SE CALIFICARÁN aquellas funciones desarrolladas en el mismo archivo que la función main.
- El código comentado NO SE CALIFICARÁ.
- Los proyectos deben obligatoriamente desarrollarse en NetBeans bajo el sistema operativo Windows. No se revisarán los proyectos desarrollados en otros sistemas operativos o IDEs.
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no muestres resultados o que estos no sean coherentes en base al 60%.
  Se tomará en cuenta en la calificación el uso de comentarios relevantes.

SE LES RECUERDA QUE, DE ACUERDO AL REGLAMENTO DISCIPLINARIO DE NUESTRA INSTITUCIÓN, CONSTITUYE UNA FALTA GRAVE COPIAR DEL TRABAJO REALIZADO POR OTRA PERSONA O COMETER PLAGIO. ESTO Y EL HECHO DE ENCONTRAR CUALQUIER ARCHIVO YA SEA .cpp O .h CON FECHA U HORA DE CREACIÓN ANTERIOR A LA EVALUACIÓN SERÁ CONSIDERADO UNA FALTA DE PROBIDAD Y POR LO TANTO AMERITARÁ LA ANULACIÓN DE LA PRUEBA.

NO SE HARÁN EXCEPCIONES ANTE CUALQUIER TRASGRESIÓN DE LAS INDICACIONES DADAS EN LA PRUEBA

- **Puntaje total**: 20 puntos.

Cuestionario:

La finalidad principal de este laboratorio es la de reforzar los conceptos contenidos en los capítulos 6 y 7 del tema: "Programación orientada a objetos" y "Operadores Sobrecargados".

Cree una **carpeta** denominada "Infracciones2021-1Lab07", dentro de ella cree dos **carpetas** denominadas "Lab07-Parte01" y "Lab07-Parte02", en estas colocará los proyectos solicitados en las preguntas 1 y 2 respectivamente. DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 3 PUNTOS DE LA NOTA FINAL.

PARTE01 (12 puntos): CREACIÓN DE LAS CLASES

Se solicita que desarrolle un proyecto **"LAB07_PREG01_CLASES"** dentro de la carpeta correspondiente, en la cual se definan clases para el manejo de los conductores y sus respectivas infracciones. Así como una serie de operadores sobrecargados que permitan manejar estas clases. A continuación, se definen las clases que serán necesarias:

➢ **Para manejar los conductores**: La clase se denominará "**Conductor**" y deberá contener lo siguiente: 1) un atributo denominado **licencia**, este atributo debe almacenar la licencia del conductor (int), 2) un atributo denominado **nombre** definido por una cadena de caracteres (char*), 3) un atributo denominado **lfaltas**, definido por un arreglo de clases de tipo FaltaCond, 4) un atributo denominado numFaltas (int) que llevará la cuenta de las faltas cometidas por el conductor, 5) un atributo denominado montoTotal de tipo double, que acumulará el monto total a pagar por las faltas.

➢ **Para manejar las faltas de un conductor**: La clase se llamará "**FaltaCond**" y deberá contener lo siguiente: 1) un atributo denominado placa definido por una cadena de caracteres (char*), 2) un atributo denominado fecha (int) que almacenará una fecha en el formato AAAAMMDD, 3) un atributo denominado codInf (int) que almacenará el código de la infracción, 4) un atributo denominado multa (double) que almacenará el monto a pagar por la multa, 5) un atributo denominado gravedad, cadena de caracteres (char*) que almacenará: "LEVE", "GRAVE" o "MUY GRAVE",.

➢ **Para manejar las infracciones cometidas**: La clase se denominará "**Falta**" y deberá contener lo siguiente: 1) un atributo denominado licencia, este atributo debe almacenar la licencia del conductor que ha cometido una falta (int), 2) un atributo denominado placa definido por una cadena de caracteres (char*), 3) un atributo denominado codInf con el código de la infracción cometida (int), 4) un atributo denominado fecha (int) que almacenará una fecha en el formato AAAAMMDD.

➢ **Para manejar las infracciones establecidas que se pueden cometer**: la clase se denominará "**Infraccion**" y deberá contener lo siguiente: 1) un atributo denominado codigo con el código de la infracción (int), 2) un atributo denominado descripcion definido por una cadena de caracteres (char*) que indicará la descripción de la falta, 3) un atributo denominado gravedad definido por una cadena de caracteres (char*) que indicará la gravedad de la falta ("LEVE", "GRAVE" o "MUY GRAVE"), 4) un atributo denominado multa (double) que almacenará el pago que debe hacerse por la multa.

"**DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS**"

Las operaciones que se permitirá realizar a través de sobrecargas de operadores se definen a continuación:

➢ **Lectura**:

- Sobrecargando el operador **>>** de modo que permita leer un conductor de un archivo. La operación **(arch >> conductor;)** involucrará un archivo y un objeto de la clase "**Conductor**". Una línea de archivo tendrá la siguiente forma:

```
63736112,ZAMORA ZAVALETA RONAL MANUEL (licencia, nombre)
```

- Sobrecargando el operador **>>** de modo que permita leer una falta cometida de un archivo. La operación **(arch >> falta;)** involucrará un archivo y un objeto de la clase "**falta**". Una línea de archivo tendrá la siguiente forma:

```
15029228,C0P-895,10/12/2019,201   (licencia, placa, fecha, código de infracción)
```

- Sobrecargando el operador **>>** de modo que permita leer una infracción establecida de un archivo. La operación **(arch >> infraccion;)** involucrará un archivo y un objeto de la clase "**Infraccion**". Una línea de archivo tendrá la siguiente forma:

```
101,Adelantar en forma indebida a otro vehículo.,Grave,316.00 (código, descripción, gravedad, multa)
```

Estas sobrecargas pueden ser definidas en una biblioteca exclusiva o como parte de la clase que manejan.

➢ **Agregación**:

- Sobrecargando el operador + de modo que permita agregar **una** falta a **un** conductor. La operación (**conductor + falta;**) colocará los datos de la falta al final de su lista **lfaltas**, modificando el número de faltas (numFaltas). No debe modificar los atributos que no están involucrados en la operación.
- Sobrecargando el operador + de modo que permita completarlos datos de algunas faltas cometidas por **un** conductor. La operación (conductor + infraccion;) colocará la gravedad y multa

en todas las faltas que coincidan con la infracción dada (infraccion). No debe modificar los atributos que no están involucrados en la operación.

- **Cálculo del monto total de infracciones cometidas**: sobrecargando el operador ++ de modo que permita calcular la suma total por las infracciones cometidas por un conductor.
- **Amnistía de infracciones**: sobrecargando el operador * de modo que permite aplicar una serie de beneficios a un conductor. La operación (**conductor * fecha;**), donde la fecha estará dada por un valor entero de la forma AAAAMMDD. La amnistía consistirá en reducir en un 25% sus infracciones graves, en un 10% sus infracciones muy graves y 50% en las infracciones leves, de aquellas infracciones que se cometieron antes de la fecha dada.
- **Impresión**: sobrecargando el operador << de modo que permita imprimir la información de un conductor. La operación (**arch << conductor;**) permitirá imprimir en un archivo de textos los datos contenidos en un objeto de tipo "**Conductor**". El formato será el siguiente:

```
Conductor   :ZAMORA ZAVALETA RONAL MANUEL
Licencia No.:63736112
============================================================
Infracciones cometidas:
------------------------------------------------------------
No.   Fecha        Placa     Infracción   Gravedad    Multa
1)    30/07/2020   T9A-930       137      Grave        316.00
2)    21/10/2019   K0D-676       302      Muy Grave   1777.50
…     …            …              …                  …
============================================================
                          Cantidad      Total
Total de Infracciones:       8         3863.10
```

El reporte debe estar perfectamente tabulado (**sin usar el carácter '\t'**).

Consideraciones:

La solución debe contemplar la elaboración del proyecto de implementación, y la prueba de las sobrecargas en el main, no hay problema si para esta labor se excede en el número de líneas.

La prueba de las sobrecargas debe ser hecha lo más simple posible pero que se muestre claramente que son correctas.

PARTE 2(8 puntos): Prueba final.

Desarrolle en la carpeta "Lab07-Parte02" un proyecto denominado "**LAB07_PREG02_CARGA**" en el cual se utilizará obligatoriamente las clases desarrolladas en la pregunta anterior. El proyecto ejecutará las tareas descritas a continuación utilizando las sobrecargas definidas anteriormente:

a) Leer los datos de los conductores contenidos en un archivo como se muestra a continuación y los coloque en un arreglo de la clase "**Conductor**":

```
63736112,ZAMORA ZAVALETA RONAL MANUEL
45043076,VEGA VILCARA CARMELA TERESA
…
```

b) Leer los datos del registro de faltas cometidas y asignárselas a cada conductor del arreglo anterior. El archivo que es similar al siguiente:

```
15029228,C0P-895,10/12/2019,201
48450395,Q5S-871,16/10/2020,151
…
```

c) Leer una a una las infracciones establecidas y completar las multas y gravedad de las faltas cometidas por los conductores del archivo que es similar al siguiente:

```
101,Adelantar o sobrepasar en forma indebida a otro vehículo.,Grave,316.00
102,No hacer señales ni tomar las precauciones para girar voltear en U.,Grave,316.00
103,Detener el vehículo bruscamente sin motivo.,Grave,316.00
…
```

d) Calcular el monto total por las infracciones cometidas por los conductores del arreglo.

e) Emitir un reporte en el que se muestren los datos de cada conductor.

f) Aplicar una amnistía a los conductores para una fecha ingresada por teclado y calcular en nuevo monto total por las infracciones.

g) Emitir un reporte en el que se muestren los datos de cada conductor con la amnistía aplicada.

Para realizar las operaciones solicitadas debe desarrollar las funciones necesarias, en una biblioteca exclusiva para estas acciones. Si necesita puede manejar una variable entera en el main para el control del número de conductores. Finalmente no puede agregar más atributos de los indicados en el enunciado pero si los métodos que sean necesarios.

Al finalizar la práctica, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este laboratorio.

Profesores del curso: Miguel Guanira
Rony Cueva

San Miguel, 18 de junio del 2021.