PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
9na práctica (tipo b)
Segundo Semestre 2023

Indicaciones Generales:

Duración: 110 minutos.

SOLO ESTÁ PERMITIDO EL USO DE APUNTES DE CLASE. NO PUEDE UTILIZAR FOTOCOPIAS NI MATERIAL IMPRESO, TAMPOCO PODRÁ EMPLEAR HOJAS SUELTAS.

- No se pueden emplear **variables globales, ni estructuras**. **No puede utilizar la clase (o el tipo de datos) string**. 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. **NO PODRÁ EMPLEAR PLANTILLAS EN ESTE LABORATORIO**
- 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 el archivo main.cpp deberá colocar un comentario en el que coloque claramente su nombre y código, **de no hacerlo se le descontará 0.5 puntos en la nota final.**
- El código comentado NO SE CALIFICARÁ. De igual manera NO SE CALIFICARÁ el código de una función si esta función no es llamada en ninguna parte del proyecto o su llamado está comentado.
- 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%.
- **Deberá mantener en todo momento el encapsulamiento de todos los atributos de las clases, así como guardar los estándares en la definición y uso de todas las clases desarrolladas.** No se considerará en la nota las clases que violen esto.
- Salvo en la sobrecarga de los operadores >> y <<, **no se podrán definir funciones (ni plantillas de funciones) No está permitido el uso de la sentencia friend.** Tampoco se podrá emplear la cláusula ***protected***.
- 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.

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

- **Puntaje total**: 20 puntos.

INDICACIONES INICIALES

Cree un proyecto de C++ en NetBeans siguiendo estrictamente las indicaciones que a continuación se detallan:

- La unidad de trabajo será **t:** (Si lo coloca en otra unidad, no se calificará su laboratorio y se le asignará como nota cero)
- Cree allí una carpeta con el nombre **"CO_PA_PN_Lab09_2023_2"** donde **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno y **PN** primer nombre (de no colocar este requerimiento se le descontará 3 puntos de la nota final). **Allí colocará los proyectos solicitados en la prueba.**

Cuestionario:

PARTE01 (10 puntos): CREACIÓN DE LAS CLASES

Se solicita que desarrolle un proyecto **"LAB09_PREG01"** dentro de la carpeta correspondiente, DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 2 PUNTOS DE LA NOTA FINAL, en la cual se declaren las clases descritas con las relaciones necesarias, que permitan manipularlas empleando herencia:

➢ **Para manejar los pedidos realizados**: La clase se denominará "**NPedido**" y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**char***) definido por una cadena dinámica de caracteres que representa el código del producto, 2) un atributo denominado **cantidad** (**int**) que representa la cantidad de productos solicitado, 3) un atributo denominado **peso** (**double**) que representa el peso de todo el pedido.

➢ **Para manejar los vehículos**: La clase se denominará "**Vehiculo**" y deberá contener lo siguiente: 1) un atributo denominado **cliente** (**int**) que representa el código del cliente al cual pertenece el vehículo, 2) un atributo denominado **placa** (**char***) definido por una cadena dinámica de caracteres, 3) un atributo denominado **maxcarga** (**double**) que representa el peso máximo que puede transportar un vehículo, 4) un atributo denominado **actcarga** (**double**) que representa el peso que está transportando el vehículo de acuerdo a la carga que lleva, recuerde que nunca puede exceder a **maxcarga**.

➢ **Para manejar los vehículos tipo camión**: La clase se denominará "**Camion**" y deberá contener lo siguiente: 1) un atributo denominado **ejes** (**int**) que representa la cantidad de ejes que tiene el vehículo, 2) un atributo denominado **llantas** (**int**) que representa la cantidad de llantas que tiene el vehículo, 3) un atributo denominado **mdeposito**, este es un STL-Map de la clase **NPedido**, donde se guardarán los pedidos que el camión puede transportar, se sabe que cada camión puede llevar 5 pedidos como máximo, por tal motivo debe usar el STL-Map para simular el depósito de la unidad, identificando con un número entero cada pedido que lleva el camión. Esta clase posee datos heredados de la clase **Vehiculo**.

➢ **Para manejar los vehículos tipo furgón**: La clase se denominará "**Furgon**" y deberá contener lo siguiente: 1) un atributo denominado **filas** (**int**) que representa la cantidad de filas que tiene el vehículo, 2) un atributo denominado **puertas** (**int**) que representa la cantidad de puertas que tiene el vehículo, 3) un atributo denominado **pdeposito**, este es un STL-List de la clase **NPedido**, donde se guardarán los pedidos que lleva el furgón. Debido al tipo de vehículo los pedidos se cargan apilados, por tal motivo la STL-List debe simular el comportamiento de una **pila**. **De no realizarlo esta clase será invalida para la calificación**. Esta clase posee datos heredados de la clase **Vehiculo**.

➢ **Para manejar los nodos de los vehículos**: La clase se denominará "**NVehiculo**" y deberá contener lo siguiente: 1) un atributo denominado **unidad,** este es un puntero de la clase **Vehiculo**.

➢ **Para manejar la flota de vehículos**: La clase se denominará "**Flota**" y deberá contener lo siguiente: 1) un atributo denominado **vflota**, este es un STL-Vector de la clase **NVehiculo**, donde se guardarán todos los vehículos, recuerde que cada cliente tiene un vehículo, tenga o no pedidos.

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

Con las clases indicas debe realizar las siguientes operaciones:

- En la clase **Flota** implementar el método **cargaflota,** que se encarga de la lectura del archivo "**Vehiculos.csv**" y cargar la información en el STL-Vector denominado **vflota**. Para la lectura de los datos correspondiente a cada vehículo debe utilizar el método polimórfico **lee**, ya que los mismos pueden ser furgones o camiones. Para diferenciar cada tipo de vehículo en el archivo correspondiente los furgones se representan con la letra "F" y los camiones con la "C".
- En la clase **Flota** implementar el método **muestracarga,** que se encarga de realizar la impresión de un archivo debidamente tabulado (**sin usar el carácter '\t'**), que muestre todos los datos de cada vehículo de acuerdo con el tipo de unidad. Para este paso debe utilizar el método polimórfico **imprime**. Antes del llenado del depósito del vehículo, debe mostrar el mensaje **"No hay pedidos para el cliente"**.

Para esta pregunta, por lo menos debe desarrollar los siguientes métodos polimórficos:

➢ **lee**: para la lectura de los datos de cada uno de los vehículos de acuerdo con su tipo.
➢ **imprime**: para la impresión de los datos de cada uno de los vehículos de acuerdo con su tipo.

Consideraciones:
Para el desarrollo de ambas preguntas debe considerar el siguiente código, con excepción del método **cargapedidos** que solo debe estar en la pregunta 2:

```
#include "Flota.h"

using namespace std;

int main(int argc, char** argv) {
   Flota Unidades;

   Unidades.cargaflota();
   Unidades.cargapedidos();
   Unidades.muestracarga();

   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

PARTE 2 (10 puntos): Proceso final.
Desarrolle un proyecto denominado **"LAB09_PREG02"** en el cual se utilizará obligatoriamente las clases desarrolladas en la pregunta anterior. El proyecto ejecutará las tareas descritas a continuación:

- Cargar el STL-Vector denominado **vflota** de acuerdo con lo indicado a la pregunta anterior.
- Ordenar los vehículos de **vflota** por el código del cliente de forma ascendente.
- Desarrollar un método denominado **cargapedidos** que pertenece a la clase **Flota**, que se encarga de leer el archivo "**Pedidos3.csv**", al obtener el código del cliente al que le pertenece el pedido, deberá buscar en STL-Vector **vflota** el vehículo del cliente y colocar el pedido en el STL-List o STL-Map según corresponda, considerando no sobrepasar la **maxcarga** al añadir el pedido, para este control, cada vez que coloca un pedido debe actualizar el atributo **actcarga**. **Para la carga de los pedidos en los vehículos debe usar el método polimórfico cargadeposito. Recuerde que los camiones solo pueden llevar 5 productos.**
- Finalmente imprimir un reporte con las unidades del STL-Vector y el contenido de cada STL-List o STL-Map, de acuerdo con el vehículo, para esta operación puede modificar el método **muestracarga** de la pregunta anterior. Para la impresión puede recorrer el STL-List o STL-Map de acuerdo con el vehículo del cliente. El reporte debe tener el siguiente formato:

```
                    REPORTE DE FLOTA
============================================================

Codigo Cliente: 12663268
Placa:          U0T-459
Carga Maxima:   1500.00
Carga Actual:   546.50
#Ejes:          3
#Llantas:       8
Lista de Pedidos:
  1   AHB.459    3       5.10
  2   YYU.726    4       6.40
  3   XSD.310    6     330.00
  4   RAH.420    3     150.00
  5   BMJ.772    1      55.00

Codigo Cliente: 13245501
Placa:          U6A-456
Carga Maxima:   300.00
Carga Actual:   294.75
#filas:         2
#puertas:       2
Lista de Pedidos:
   KJX.387    5      25.00
   QIL.029    2       6.00
   BXY.119    5     225.00
   ZWF.661    1       5.00
   HFC.845    6      30.00
   VXS.438    3       0.75
   AWB.345    1       3.00
...
```

Se recomienda revisar los archivos que servirán para la lectura de datos, los cuales se describen a continuación:

Pedidos3.csv

```
50375303,JXD.139,6,120
50375303,CRU.009,5,200
22777006,YYK.309,3,165
42157219,OTS.581,5,2.5
13245501,AWB.345,1,3
…
```

Cliente, Cod del Producto, cantidad, peso del pedido.

Vehiculos.csv

```
F,79464412,K0D-676,300,1,3
F,16552775,S7E-946,300,2,3
C,20864087,O5L-856,1000,2,6
C,94326707,U3F-754,1000,2,8
…
```

Tipo de Vehículo, Cliente, Placa, Máxima carga, Filas/Ejes, Puertas/Llantas.

Recuerde que si no usa polimorfismo la respuesta no será válida, así mismo siempre debe mantener el encapsulamiento por tal motivo cada operación debe ser desarrollada donde le corresponde, por ejemplo si va a leer o imprimir un pedido esta operación la debe realizarse en la clase que le corresponde, en este caso NPedido.

Al finalizar la práctica, comprima la carpeta dada en las indicaciones iniciales 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.

Profesores del curso: Rony Cueva
Erasmo Gómez
Miguel Guanira

San Miguel, 17 de noviembre del 2023.