PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 2
1ra práctica (tipo b)
Primer Semestre 2025

Indicaciones Generales:

- Duración: 110 minutos.

NO SE PERMITE EL USO DE APUNTES DE CLASE, FOTOCOPIAS NI MATERIAL IMPRESO

- No se pueden emplear **variables globales, NI OBJETOS** (con excepción de los elementos de **iostream, iomanip y fstream**). **NO PUEDE UTILIZAR LA CLASE string**. Tampoco se podrán emplear las funciones malloc, realloc, memset, strtok o strdup, 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%.
- 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_Lab01_2025_1"** 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:

La finalidad principal de este laboratorio es la de reforzar los conceptos contenidos en el capítulo 1 del curso: "Funciones y alcance de variables". En este laboratorio se desarrollará **una biblioteca estática de funciones** en la que se implementen sobrecargas de operadores y funciones que le permitan solucionar el problema planteado.

En la carpeta solicitada anteriormente, cree **dos carpetas** denominadas **"Parte1_CrearBiblioteca"** y **"Parte2_UsarBiblioteca"**, allí colocará los proyectos solicitados en las partes 1 y 2 respectivamente de este laboratorio. DE NO COLOCAR ALGUNO DE ESTOS REQUERIMIENTO SE LE DESCONTARÁ 3 PUNTOS DE LA NOTA FINAL. **NO SE HARÁN EXCEPCIONES.**

PARTE 01 (12 puntos): CREACIÓN DE LA BIBLIOTECA ESTÁTICA

En el lenguaje C++ se define una biblioteca de funciones denominada <string> en la que se define una clase denominada también string. Esta clase permite definir objetos para manejar cadenas de caracteres que,

a diferencia del C estándar, son "más fáciles de manipular y más seguras". Sin embargo, como la mayoría comprende, esta clase no aparece de la nada, alguien se tomó un tiempo para diseñarla e implementarla en base a elementos más simples del lenguaje.

Se solicita que desarrolle una biblioteca estática en la cual se defina un tipo de dato y una serie de funciones y operadores sobrecargados que permitirá manejar cadenas de caracteres de una manera más simple y segura.

El tipo de dato que manejará la estructura será el siguiente:

```
struct CadenaDeCaracteres {
   char *cadena;
   int longitud;
   int capacidad;
};
```

Donde "longitud" y "capacidad" deberán actualizarse en cada operación que se realice con la estructura. La capacidad debe contener el espacio de memoria asignado al arreglo "cadena" y longitud deberá contener la cantidad de caracteres válidos que contiene. No debe modificar los identificadores ni los tipos de datos de esta estructura, tampoco podrá eliminar o agregar campos, de hacerlo se le calificará la prueba con la mitad del puntaje. Tampoco podrá definir estructuras adicionales en toda la prueba.

Las operaciones que la biblioteca deberá realizar, a través de sobrecargas de operadores y funciones, se definen a continuación:

Inicialización:

- ✓ Sobrecargando el operador **!** de modo que permita inicializar una variable de tipo **struct CadenaDeCaracteres**. La operación **!cadena** deberá inicializar los campos de la estructura en nulo o cero respectivamente.

Asignación:

- ✓ Sobrecargando el operador **<=** de modo que permita asignar una cadena de caracteres del tipo definido por el lenguaje *C/C++* a una variable de tipo **struct CadenaDeCaracteres,** donde en las operaciones **cadena <= cad;** o **cadena <= "Valentina Gutierrez",** el primer operando será de tipo **struct CadenaDeCaracteres** y el segundo de tipo **char*.**
- ✓ Sobrecargando el operador **<=** de modo que permita asignar un espacio de memoria a una variable de tipo **struct CadenaDeCaracteres**. En la operación **cad <= 10;** se asignará un espacio de memoria al campo **cadena**, la variable **cad** quedará vacía.
- ✓ Sobrecargando el operador **<=** de modo que permita copiar una variable del tipo **struct CadenaDeCaracteres** a otra del mismo tipo. La operación **cadena1 <= cadena2,** ambas de tipo **struct CadenaDeCaracteres**, tomará el contenido del campo cadena de segundo operando y lo copiará al mismo campo del primer operando.

En el caso que el campo cadena sea nulo, deberá asignar un espacio exacto de memoria, si el campo cadena está apuntando a un espacio válido, se copiará en ella la cadena si el espacio es suficiente para contener la cadena de caracteres a copiar, sino deberá gestar un nuevo espacio de memoria (exacto). La operación deberá liberar los espacios de memoria que no se usarán.

Concatenación

- ✓ Sobrecargando el operador **+=** de modo que permita concatenar una cadena de caracteres del tipo definido por el lenguaje *C/C++* a una variable de tipo **struct CadenaDeCaracteres.** La operación **cadena += "Naomi Guzman";** concatenará la cadena dada por el segundo operando al campo **cadena** del primer operando.
- ✓ Sobrecargando el operador **+=** de modo que permita concatenar la cadena contenida en una variable de tipo **struct CadenaDeCaracteres** a otra del mismo tipo. La operación **cadena1 += cadena2,** ambas de tipo **struct CadenaDeCaracteres**, tomará el contenido del campo cadena de segundo operando y lo concatenará al mismo campo del primer operando.

La concatenación deberá devolver false si la cadena del primer operando es nula o vacía y no realizará la concatenación, devolverá true si la operación se pudo realizar, si el

espacio de memoria de la cadena del primer operando no puede contener la concatenación, deberá gestar un nuevo espacio de memoria (exacto).

Comparación

- ✓ Definiendo la función **compare** de modo que permita comparar una variable de tipo **struct CadenaDeCaracteres** con una cadena de caracteres del tipo definido por el lenguaje C/C++. La operación **compare(cadena, "Ana Rojas")** comparará el campo **cadena** del primer operando con el segundo operando. Devolverá cero si las cadenas son iguales, un valor mayor que cero si la cadena del primer parámetro es mayor que la del segundo y un valor menor que cero si es menor.
- ✓ Sobrecargando los operadores ==, > y < que realizarán la misma tarea que el anterior pero la comparación devolverá **true** si se cumple la condición y **false** si no.
- ✓ Definiendo la función **compare** de modo que permita comparar dos variables de tipo **struct CadenaDeCaracteres**. La operación **compare(cadena1, cadena2)** comparará los campos **cadena** del ambos operandos. Devolverá cero si las cadenas son iguales, un valor mayor que cero si la cadena del primer parámetro es mayor que la del segundo y un valor menor que cero si es menor.
- ✓ Sobrecargando los operadores ==, > y < que realizarán la misma tarea que el anterior pero la comparación devolverá **true** si se cumple la condición y **false** si no.

LAS COMPARACIONES NO DEBEN DISTINGUIR ENTRE MAYÚSCULAS Y MINÚSCULAS, ESTO ES 'A' = 'a', 'B' = 'b', etc. NO DEBE MODIFICAR EN ESTE SENTIDO LAS CADENAS DE ORIGEN.

Lectura:

- ✓ Sobrecargando el operador >> de modo que permita leer una palabra de un archivo de textos. La operación **(arch >> cadena;)** involucrará una variable de archivo y una variable de tipo **struct CadenaDeCaracteres**. La sobrecarga deberá devolver cero si pudo leer la cadena, 1 si no pudo leerla porque llegó al fin del archivo y -1 si la cadena no es una palabra porque no empieza con una letra del alfabeto inglés.

Intercambio:

- ✓ Sobrecargando el operador **&&** de modo que permita intercambiar los contenidos de dos cadenas de tipo **struct CadenaDeCaracteres**. La operación **cadena1 && cadena2** realizará en intercambio entre las dos cadenas. La operación no devuelve resultados.

Impresión

- ✓ Sobrecargando el operador << de modo que permita imprimir en archivo de textos el contenido del campo cadena de una variable de tipo **struct CadenaDeCaracteres**.

EN TODOS LOS CASOS SE PODRÁ EMPLEAR EL LLAMADO A UNA SOBRECARGA EN LA IMPLEMENTACIÓN DE OTRA, POR TAL MOTIVO, NO SE CALIFICARÁN AQUELLAS SOBRECARGAS QUE COPIEN O DUPLIQUEN EL CÓDIGO DE OTRAS.

Consideraciones:

La solución debe contemplar la elaboración de:

1) Un proyecto de implementación y prueba de las sobrecargas, denominado "**Bibioteca_20225_1_Fuentes**", la biblioteca tendrá el mismo nombre que el proyecto.

2) Un proyecto que genere la biblioteca estática (.a) denominado "**Bibioteca_2025_1_Compilada**".

3) Un proyecto donde se pruebe la biblioteca ya compilada "**Biblioteca_Compilada_2025_1_Prueba**". La prueba de las sobrecargas para el primer y tercer proyecto deben ser hecha lo más simple posible pero que muestre claramente que son correctas Los tres proyectos deberán colocarse en la carpeta "**Parte1_CrearBiblioteca**".

NO SE DEBE SOLUCIONAR AQUÍ EL PROBLEMA DE LA PARTE 2, DE HACERLO NO SE LE CALIFICARÁ LA SEGUNDA PARTE DE ESTE LABORATORIO.

EN LAS PRUEBAS PARA LOS PROYECTOS 1 Y 3 SE PUEDE HACER DIRECTAMENTE EN LA FUNCIÓN MAIN, SE PUEDE HACER USO DE "HARD CODE" Y SE PUEDE SOBREPASAR LA CANTIDAD EL NUMERO DE LÍNEAS, SOLO EN ESTA PARTE DE LA PRUEBA.

EN LA BIBLIOTECA ESTÁTICA, NO PUEDE DESARROLLAR OTRAS FUNCIONES O SOBRECARGAS ADICIONALES A MENOS QUE ESTÉN DIRECTAMENTE RELACIONADAS A LAS SOBRECARGAS SOLICITADAS.

PARTE 2 (8 puntos): REUTILIZACIÓN DE LA BIBLIOTECA ESTÁTICA.

Desarrolle un proyecto denominado **"AplicacionConBibEstatica"** (dentro de la carpeta **"Parte2_UsarBiblioteca"**) en el cual se enlazará y utilizarán obligatoriamente las sobrecargas de la biblioteca estática compilada **(.a) desarrollada en la Parte 1**.

LA PREGUNTA NO SE EVALUARÁ SI EN ESTE PROYECTO NO SE USAN (O NO SE VE QUE USAN) LAS SOBRECARGAS IMPLEMENTADAS EN LA BIBLIOTECA COMPILADA (.a).

Se tiene un archivo de textos como el que se muestra a continuación:

```
12270502  Cueva      Fuentes   Cinthia Delia  129  P7T-999
12443643   Lee  Serrano   Rosario  140  R8H-409
…
```

En cada línea se encuentra el dato de un conductor que ha cometido una infracción de tránsito, primero aparece el DNI del conductor, sus apellidos y nombres, el código de la infracción que cometió y la placa del automóvil que estaba conduciendo. Observe que los datos están separados por uno o más espacios en blanco.

El programa deberá leer el archivo y colocar los datos en tres arreglos (**`int *dni`, `struct CadenaDeCaracteres *conductor` y `struct CadenaDeCaracteres *placa`**). En el caso del nombre del conductor se debe guardar el nombre completo del conductor separando cada palabra con un guion bajo, por ejemplo: Cueva_Fuentes_Cinthia_Delia. Luego el programa deberá ordenar los arreglos de manera descendente por el nombre del conductor (por única vez en el semestre se permitirá ordenar por el método de intercambio). Finalmente deberá mostrar en un archivo de textos un reporte similar al siguiente:

```
                    REPORTE DE INFRACTORES DE TRANSITO
================================================================================
DNI        INFRACTOR                      PLACA
--------------------------------------------------------------------------------
12270502   Cueva_Fuentes_Cinthia_Delia    P7T-999
12443643   Lee_Serrano_Rosario            R8H-409
…          …                              …
```

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
Andrés Melgar
Erick Huiza
Miguel Guanira

San Miguel, 11 de abril del 2025.