PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 2

7ma práctica (tipo b)
Segundo 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** .
- **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.
- TAMPOCO SE PODRÁ EMPLEAR LA CLÁUSULA protected NI LA CLÁUSULA friend, DE HACERLO NO SE LE CALIFICARÁN LAS CLASES INVOLUCRADAS.

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 CLion 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_Lab07_2025_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 el proyecto solicitado en la prueba.**

Cuestionario:

La finalidad principal de este laboratorio es la de reforzar los conceptos contenidos en el capítulo 3 del curso: "Programación Orientada a Objetos". En este laboratorio se trabajará con Herencia de Objetos.

Deberá elaborar un proyecto denominado "StreamersHerencia" y en él desarrollará el programa que dé solución al problema planteado. DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 3 PUNTOS DE LA NOTA FINAL. **NO SE HARÁN EXCEPCIONES**

NO PODRÁ EMPLEAR ARREGLOS DE MÁS DE UNA DIMENSIÓN

NO PUEDE MANIPULAR UN PUNTERO CON MÁS DE UN ÍNDICE

LOS ARCHIVOS SOLO SE PUEDEN LEER UNA VEZ

Se te ha contratado para apoyar a un pequeño estudio en el análisis de **métricas de creadores de contenido (streamers)**. El objetivo es desarrollar un **sistema de gestión de streamers** aplicando los principios de **herencia y encapsulamiento** en programación orientada a objetos.

En este sistema, deberás distinguir entre dos tipos principales de usuarios:

- **Streamers de Pago**, que cuentan con acceso a herramientas avanzadas y pueden visualizar métricas detalladas como horas vistas, nivel de interacción con la audiencia, ingresos generados y calidad técnica de sus transmisiones.

- **Streamers Gratuitos**, que solo acceden a métricas básicas de su actividad y reciben un mensaje de *upsell* invitándolos a adquirir la licencia premium para obtener estadísticas avanzadas ("Adquiere la licencia para ver estadísticas avanzadas").

A su vez, el sistema deberá manejar distintos tipos de **métricas**, cada una representando un aspecto diferente del desempeño de los streamers:

- **Métrica Básica**: mide la actividad general del canal, como las horas transmitidas y el promedio de espectadores.

- **Métrica Engagement**: evalúa la interacción del público, considerando la cantidad de mensajes en el chat, usuarios únicos activos y clips creados durante las transmisiones.

- **Métrica Calidad**: analiza el rendimiento técnico del stream, como el bitrate promedio, la proporción de cuadros perdidos y los cuadros por segundo (FPS) alcanzados.

Tu tarea consistirá en diseñar las clases necesarias, leer los datos desde **archivos de texto** (o CSV) y generar reportes que muestren las estadísticas correspondientes según el tipo de streamer. Este ejercicio busca evaluar tu dominio de **herencia, encapsulamiento, sobrecarga, y manejo de arreglos** en C++, manteniendo una arquitectura limpia, reutilizable y con responsabilidad clara por tipo de objeto.

Se tienen dos archivos del tipo CSV, los cuales se describen a continuación:

streamers.csv

```
Gratuito,174920,xQcOW,PUBG,3246298,2025-10-17
Pagado,839427,summit1g,CODWarzone,5310163,2025-08-07
…
```

tipo, id_stream, cuenta, categoría, seguidores.
Luego puede venir, dependiendo del tipo:
Gratuito : fecha_fin_trial
Pagado : fecha_inicio_plan

metricas.csv

```
ENGAGEMENT,573984,161,Q&A largo,2025-08-07,2025-12-31,1586,571,27
BASIC,263905,44,Sesión Games,2025-11-26,2025-12-31,6.5,1116
QUALITY,314875,3033,Bitrate consistente,2025-07-27,2025-12-31,7586,0.063,48
…
```

tipo, id_stream, id_metrica, descripcion, fecha_calculo, fecha_expiracion.
Luego puede venir, dependiendo del tipo:
BASIC : horas_transmitidas, espectadores_promedio
ENGAGEMENT : mensajes_chat, usuarios_unicos_chat, clips_generados
QUALITY : bitrate_promedio_kbps, porcentaje_frames_perdidos, fps_promedio

PARTE 1 (3 puntos)

Construye una clase **Metrica**, que soporte los atributos generales de cualquier métrica asociada a un streamer.
Para ello, añade los siguientes atributos privados:

- `id:` entero (`int`) que almacena el id de la métrica.
- `descripcion`: cadena (char*) que almacena la descripción de la métrica.
- `fecha_calculo`: entero (`int`) que almacena la fecha en la que se calculó la métrica (formato AAAAMMDD).
- `fecha_expiracion`: entero (`int`) que almacena la fecha límite de validez de la métrica (formato AAAAMMDD).
- `estado`: booleano (`bool`) que indica si la métrica se encuentra **activa** (`true`) o **expirada** (`false`).

Deberás implementar **todos los constructores, destructores, getters y setters** necesarios para esta clase.
Además, implementa los siguientes métodos:

- `leer(ifstream&)`: permite **leer la información de una métrica desde un archivo CSV** y almacenarla en la clase.

- `imprimir(ofstream&) const`: muestra los datos de la métrica en una sola línea bien formateada (ya sea en archivo o en consola), de modo que la información sea clara y fácil de interpretar.

Extensión de la clase Metrica

Crea tres clases derivadas de `Metrica`, cada una representando un tipo específico de métrica, con sus propios atributos de dominio:

1. **MetricaBasica:** `double horas_transmitidas, int espectadores_promedio.`
   Representa la actividad general del canal (tiempo de transmisión y promedio de audiencia).

2. **MetricaEngagement:** `int mensajes_chat, int usuarios_unicos_chat, int clips_generados.`
   Evalúa el nivel de interacción e interés de la comunidad durante las transmisiones.

3. **MetricaCalidad:** `int bitrate_promedio_kbps, double porcentaje_frames_perdidos, int fps_promedio`
   Mide la calidad técnica del stream, su estabilidad y fluidez.

Cada clase hija deberá:

- Incluir **constructores, destructores y getters/setters propios**.

- Reutilizar los métodos `leer` e `imprimir`, extendiéndolos o sobrecargándolos para incluir los nuevos atributos.

- Mantener la coherencia con el atributo `estado`: al leer o actualizar una métrica, si la fecha actual es mayor que `fecha_expiracion`, el estado debe marcarse automáticamente como

inactivo (`false`) caso contrario activo (`true`).

PARTE 2 (3 puntos)

Construye una clase **Streamer,** que soporte los atributos de cada streamer.

Para ello añade los siguientes atributos:

1) **id,** entero (`int`) que almacena el id de la métrica..
2) **cuenta**, es un `char*` (cadena en memoria dinámica) que almacena el nombre/cuenta del streamer y se usa para mostrar y ordenar por nombre.
3) **n_seguidores**, es un entero (`int`) que contiene la cantidad total de seguidores del streamer registrada en la data.
4) **categoria**, es un `char*` (cadena en memoria dinámica) que indica la categoría principal del streamer (p. ej., "Minecraft", "Valorant", "JustChatting").

No olvides crear **todos** los constructores, destructores, getters y setters para esta clase.

Además, crea los siguientes métodos:

1) **leer**, un método que permite leer la información de un streamer de un archivo csv y alojarla en la clase.
2) **imprimir**, un método que permite mostrar la información de un streamer, ya sea en un archivo o en consola, en una línea bien definida y con los espacios requeridos para entender correctamente la información.

Extensión de la clase Streamer

Crea dos clases derivadas de **Streamer**, cada una representando un tipo específico de cuenta con acceso diferenciado a métricas y con sus propios atributos/colecciones internas:

StreamerGratuito

Representa a los streamers sin licencia de pago. Acceden solo a **métricas básicas** y muestran mensaje de *upsell*.

Atributos adicionales (privados):

- `fecha_fin_trial`
- `MetricaBasica* metricas_basicas`
- `int cantidad_metricas_basicas`
  Arreglo dinámico con métricas de tipo **Métrica Básica** (actividad general).

StreamerPago

Representa a los streamers con licencia de pago. Acceden a métricas avanzadas y reportes detallados.

Atributos adicionales (privados):

- `fecha_inicio_plan`
- `MetricaBasica* metricas_basicas`
- `int cantidad_metricas_basicas`
- `MetricaEngagement* metricas_engagements`
- `int cantidad_metricas_engagement`
- `MetricaCalidad* metricas_calidades`
- `int cantidad_metricas_calidades`

PARTE 2 (14 puntos)

Construye una clase ***GestorStreamers***, Esta clase te permitirá cargar la información necesaria y gestionar los reportes necesarios del sistema. Debe implementar el constructor por defecto y los getters y setters correspondientes.

Para ello añade los siguientes atributos:

1) **streamers_gratuitos**, es un arreglo dinámico de **StreamerGratuito**.
2) **streamers_pagados**, es un arreglo dinámico de **StreamerPagado**.
3) **cantidad_gratuitos**, un entero que lleve la cantidad de streamers gratuitos.
4) **cantidad_pagados**, un entero que lleve la cantidad de streamers pagados.

Además crea los siguientes métodos:

1) **cargar_datos_streamers(const char* file_name)**: Lee la información contenida en el archivo indicado por file_name y carga los datos de todos los streamers del sistema, inicializando los arreglos dinámicos streamers_gratuitos y streamers_pagados.
2) **actualizar_datos_metricas(const char* file_name)**: Actualiza las métricas correspondientes a cada streamer utilizando la información proporcionada en el archivo especificado por file_name.

   Debe identificar correctamente a qué streamer pertenece cada conjunto de métricas (gratuito o pagado) y actualizar su información.

3) **generar_reportes**(): Genera un **reporte individual para cada streamer**, mostrando su información básica junto con las métricas activas. El formato del nombre del reporte debe ser el siguiente:

   /Reportes/<tipo_stream>/<**cuenta**>_<**categoria**>_<**fecha**>.**txt**

   Nota1: tipo_stream puede ser "Gratuito" o "Pagado"

   Nota2: fecha puede ser fecha_fin_trial o fecha_inicio_plan, segun corresponda.

   Nota3: Puede convertir un int a char* usando una combinación de los métodos to_string y c_str.

   to_string(entero).c_str();

```
================================================================================
                          REPORTE DE STREAMERS
================================================================================

[ST] CUENTA: CanalEpsilon      | SEGUIDORES:  152340 | CATEGORÍA: JustChatting  | TIPO: StreamerPago
--------------------------------------------------------------------------------
[BASIC] HORAS_TOT:  12.70 | VIEWERS_PROM_GLOBAL:  482 | ACTIVAS:  3 | EXPIRADAS:  1
   Código   Fecha Calc.  Expira      Estado     Horas  Viewers      Descripción
   MB-0003  20250620     20251231    ACTIVA     4.20    515m        Sesión Música
   MB-0004  20250625     20251231    ACTIVA     3.50    450m        Sesión Juegos
   MB-0005  20250415     20251001    EXPIRADA   5.00     410m       Sesión Ant.
--------------------------------------------------------------------------------
[ENGAGE] CHAT_MSGS:  3600 | CHATTERS_UNICOS:  680 | CLIPS:  55 | ACTIVAS:  2 | EXPIRADAS:  0
   Código   Fecha Calc.  Expira      Estado     Msgs   Chatters    Clips    Descripción
   ME-0102  20250620     20251231    ACTIVA     1800    360        31       Chat activo
   ME-0103  20250701     20251231    ACTIVA     1800    320        24       Reacción evento
--------------------------------------------------------------------------------
[QUALITY] BITRATE_PROM: 6200 kbps | DROP%_PROM:  0.04 | FPS_PROM:  60 | ACTIVAS: 2 | EXPIRADAS: 0
   Código   Fecha Calc.  Expira      Estado     Bitrate  Drop%      FPS      Descripción
   MQ-3001  20250620     20251231    ACTIVA     6200     0.02       60       Calidad estable
   MQ-3002  20250701     20251231    ACTIVA     6200     0.06       59       Picos leves
--------------------------------------------------------------------------------
[TOTAL] METRICAS ACTIVAS:  6 | MÉTRICAS EXPIRADAS:  1 | FECHA DE REPORTE: 2025-11-07
================================================================================
```

Modelo reporte Streamer Pagado

```
================================================================================
                              REPORTE DE STREAMERS
================================================================================
[ST] CUENTA: CanalBeta        | SEGUIDORES:   20400 | CATEGORÍA: JustChatting  | TIPO: StreamerGratuito
--------------------------------------------------------------------------------
[BASIC] HORAS_TOT:   3.50 | VIEWERS_PROM_GLOBAL:   260 | ACTIVAS:  1 | EXPIRADAS:  1
    Código      Fecha Calc.     Expira          Estado      Horas   Viewers   Descripción
    MB-0008     20250610        20251231        ACTIVA      1.80     260M     Charla semanal
    MB-0009     20250320        20250901        EXPIRADA    1.70     240M     Sesión antigua
--------------------------------------------------------------------------------
[UPSELL] Funcionalidad Pro. Adquiere la licencia para ver estadísticas avanzadas.
--------------------------------------------------------------------------------
[TOTAL] MÉTRICAS ACTIVAS:  1 | MÉTRICAS EXPIRADAS:  1 | FECHA DE REPORTE: 2025-11-07
================================================================================
```

Modelo reporte Streamer Gratuito

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: Miguel Guanira, Andrés Melgar, Rony Cueva, Eric Huiza, Erasmo Gómez

San Miguel, 07 de noviembre del 2025.