Algoritmos y Programación II

Trabajo práctico Nº 4:
Trabajo grupal

“Listas con listas”

Objetivo

Desarrollar un programa orientado a objetos en C++ que maneje listas que a su vez contienen listas.

Datos de entrada

Los datos de entrada serán dos archivos de texto *películas_vistas.txt* y *películas_no_vistas.txt.*

Ambos archivos tienen el mismo formato: con un espacio en blanco separando la lista de actores / actrices y con una línea en blanco separando cada película:

Nombre_pelicula
Género
Puntaje
Director/a
actor/actriz_1 actor/actriz_2… actor/actriz_n

Ejemplo:
Mi obra maestra
Comedia negra
8
Gaston Duprat
Luis_Brandoni Lucas_Aranda Raúl_Arévalo Mahmoud_Azim Daniela_Katz Guillermo_Francella Andrea_Frigerio

I, como Icaro
Drama Suspenso
10
Henri Verneuil
Yves_Montand Michel_Albertini Roland_Amstutz Jean-Pierre_Bagot

Jack and Jill
Comedia
3
Dennis Dugan
Adam_Sandler Al_Pacino Katie_Holmes

Carmen y Lola
Drama
7
Arantxa Echevarria
Zaira_Romero Rosy_Rodriguez Rafaela_Leon Carolina_Yuste

Cementerio de animales
Terror
7
Kevin Kolsch
Jason_Clarke Amy_Seimetz John_Lithgow

Requisitos

La aplicación debe leer estos archivos y generar dos listas: películas_vistas y películas_no_vistas. Luego ofrecerá un menú para mostrarlas por separado. Además, debe elaborar y mostrar una tercera lista: películas_recomendadas. Esta lista se armará de la siguiente forma:

- Por cada película vista se tomará: el género, el director, y los actores.
- Comparando en las no vistas, si [**hay coincidencia en el género y** (**en el director o por lo menos uno de los actores**)] **o** [**el puntaje es 7 o más]**, se toma la película como recomendada.

Consideraciones

- La aplicación debe estar completamente orientada a objetos, admitiendo solo el main como función libre.

- Los archivos están bien formados pero
    - El archivo de películas vistas podría no existir. Tener en cuenta que, en ese caso, las recomendaciones solo se harán por puntaje.
    - El archivo de películas no vistas debe existir, caso contrario se debe lanzar una excepción cerciorándose de que no quede memoria sin liberar.

- Las listas deben ser dinámicas.

- Las listas deben utilizar templates.

- La cantidad de actores que intervienen en una película es variable.

- No se pueden utilizar objetos de STL.

- Para desarrollar el trabajo grupal a distancia es obligatorio utilizar alguno de estos sistemas de control de versiones: GitHub o GitLab.

Qué se evalúa

- Buenas prácticas: código, modularización, comentarios, claridad
- POO
- Funcionalidad
- Uso de templates

- Memoria dinámica
- Interfaz de usuario
- Pre y pos condiciones
- UML
- Uso de excepciones
- Participación en el sistema de control de versiones

Normas de entrega

El trabajo se desarrollará en equipos de cuatro personas.
Deben entregar:

- Documentación
    - Diagrama de clases / relación de clases UML.
- Código fuente
    - En los archivos .h van las pre y poscondiciones.
- Dos archivos para testeo (vistas y no vistas).

La fecha de entrega vence el lunes 6/7 a las 23.55hs

Puntos: 20