Tecnológico de Monterrey

Situación Problema: Modelado de Servicios de Streaming en C + +.

Programación Orientada a Objetos (TC1030).

Grupo 326.

Santiago Ramírez Niño - A01665906.

Alejandro Ignacio Vargas Cruz - A01659714.

Profesor: Artemio Urbina García.

Miércoles, 12 de junio de 2024.

ÍNDICE.

INTRODUCCIÓN.

La Programación Orientada a Objetos es un modelo de programación en el que el

software se organiza según el tipo de datos y de objetos, y sus funciones con cierta

lógica computacional. Dichos objetos son campos de datos con propiedades y comportamientos únicos y manipulables por los programadores y revelan mucha información acerca de un objeto de estudio en particular, ya que actúan como una especie de datos sobre los que se pueden realizar métodos que emplean atributos de los mismos, propiedades y acciones de los objetos que se clasifican según sus clases a las que pertenecen (su clasificación o familia de objetos). Dichas clases son el molde a seguir para cada uno de los objetos en el modelado, ya que todos y cada uno de ellos deben cumplir con los criterios de su clase para que sean manejables; dichas clases, al tratar de recrear la realidad, se relacionan con otras para interactuar en el entorno de programación mediante relaciones como lo son la herencia (una clase es un tipo de otra clase) o composición (una clase contiene otra clase), entre muchas otras.

El paradigma de la POO es un enfoque de programación que organiza el diseño de software según los propios objetos y no en las funciones o la lógica computacional como tal; y éstos son las instancias de clases que definen la esencia de los objetos y su comportamiento marcado en el program que contienen datos (atributos o propiedades) y códigos (funciones o métodos). Se basa en los principios de Abstracción, Encapsulamiento, Herencia y Polimorfismo.

- Abstracción: Simplificación de un sistema complejo para mostrar sólo los detalles relevantes en un programa.
- Encapsulamiento: Agrupación de todos los atributos y métodos de una clase dentro de una unidad del objeto.
- Herencia: Reutilizar código para que otras clases (derivadas) obtengan la misma estructura de otra clase (base)
- Polimorfismo: Objetos de diferentes clases pueden ser tratados en un mismo método de diferentes maneras. Los objetos y funciones pueden recibir un mismo mensaje y tener diferentes comportamientos.

Uno de los principales beneficios de la POO es que busca modelar de la manera más precisa posible el mundo real, que consiste en objetos con ciertas características y que cumplen ciertas acciones. Es por ello que es ampliamente utilizada en modelos computacionales, diseño web, desarrollo de aplicaciones móviles, de videojuegos, bases de datos, etc.

En este caso, se obtiene una moderación de bases de datos en servicios de streaming, consistentes en videos que pueden ser películas o episodios que componen una serie de televisión. Este tipo de bases de datos son interesantes al modelar el contenido de estos servicios, y tenerlos organizados y a la mano después de filtrarlos de un amplio catálogo, y mostrar su información en pantalla.

DESARROLLO.

a) Diseño del Proyecto.

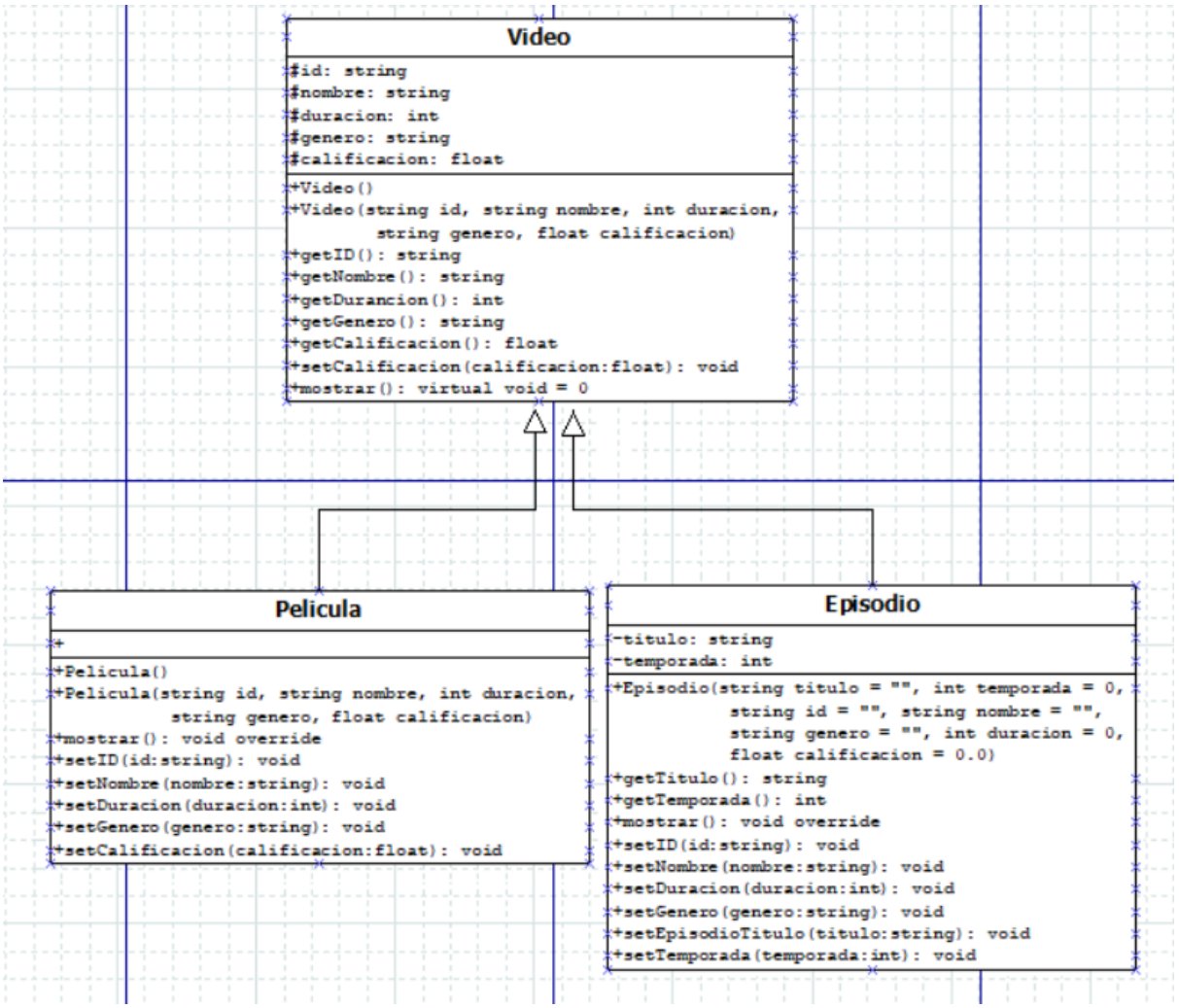
El proyecto se basa en el paradigma de la programación orientada a objetos (POO), que permite modelar entidades del mundo real (como películas y episodios) como objetos en el código. Esto facilita la comprensión del código y su mantenimiento.

Las clases Pelicula y Episodio son las principales del proyecto. Ambas heredan la clase base Video, que contiene atributos y métodos comunes a todas las películas y episodios, como el nombre, la duración y la calificación. Esto es un ejemplo de herencia, un principio de la POO que permite crear nuevas clases a partir de clases existentes, heredando sus atributos y métodos. La herencia reduce la duplicación de código y facilita el mantenimiento.

El polimorfismo es otro principio de la POO que se utiliza en este proyecto. Gracias al polimorfismo, podemos realizar, por ejemplo, sobreescritura de métodos, lo cual resulta de mucha utilidad al momento de implementar un método en la clase base, y adaptarlo en las clases derivadas para poder resolver las necesidades particulares de cada una; esto facilita el mantenimiento del código.

Decidimos realizar esta distribución de clases, ya que el proyecto giraba en torno a un servicio de streaming que tenía dos tipos de videos, películas y series. Es por esto que definimos la clase base Video, y las clases derivadas Película y Episodio, que al final, son los dos tipos de video que necesitamos. Decidimos no implementar una clase Serie, ya que una serie la podríamos expresar como un arreglo de episodios, y dicha estructura la creamos al momento de cargar nuestros archivos a la aplicación.

b) Diagrama UML.



c) Ejecución.

La ejecución del programa main refleja lo siguiente:

```
PS C:\Users\AlexN\poo_2do\Situacion_Problema> make
g++ -c -g -Wall --pedantic main.cpp
g++ -c -g -Wall --pedantic CSVLoader.cpp
g++ -c -g -Wall --pedantic Episodio.cpp
g++ -c -g -Wall --pedantic Pelicula.cpp
g++ -c -g -Wall --pedantic Video.cpp
g++ -g -Wall --pedantic main.o CSVLoader.o Episodio.o Pelicula.o Video.o -o main
```

```
PS C:\Users\AlexN\poo_2do\Situacion_Problema> ./main.exe

1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion:
```

```
Selecciona una opcion: 1
movies.csv tiene: 10 entradas
Cargando archivo: movies.csv
series.csv tiene: 50 entradas
Cargando archivo: series.csv
Archivos de datos cargados de manera exitosa!
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 2

Quiere ver los videos en base a al genero a en base a la calificacion?
a) Genero
b) Calificacion

Ingresa tu eleccion (la letra correspondiente): a
```

```
Ingresa tu eleccion (la letra correspondiente): a

Opciones de genero:
 1. Drama
 2. Accion
 3. Misterio

Selecciona el genero (Ingresa el numero correspondiente): 1
ID Pelicula: P4 | Nombre Pelicula: Before Sunrise | Duracion Pelicula: 101 | Genero Pelicula: Drama | Calificacion Pelicula: 4
ID Pelicula: P7 | Nombre Pelicula: Pelicula 7 | Duracion Pelicula: 135 | Genero Pelicula: Drama | Calificacion Pelicula: 2
ID Pelicula: P10 | Nombre Pelicula: Pelicula 10 | Duracion Pelicula: 100 | Genero Pelicula: Drama | Calificacion Pelicula: 5
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 1 | Temporada
: 1 | Calificacion Episodio: 1
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 2 | Temporada
: 1 | Calificacion Episodio: 2
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 3 | Temporada
: 1 | Calificacion Episodio: 3
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 4 | Temporada
: 1 | Calificacion Episodio: 4
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 5 | Temporada
: 1 | Calificacion Episodio: 5
ID Serie: S4 | Nombre Serie: Serie 4 | Duracion Episodio: 40 | Genero Episodio: Drama | Titulo Episodio: Episodio 1 | Temporada
: 1 | Calificacion Episodio: 1
ID Serie: S4 | Nombre Serie: Serie 4 | Duracion Episodio: 40 | Genero Episodio: Drama | Titulo Episodio: Episodio 2 | Temporada
: 1 | Calificacion Episodio: 2
ID Serie: S4 | Nombre Serie: Serie 4 | Duracion Episodio: 40 | Genero Episodio: Drama | Titulo Episodio: Episodio 3 | Temporada
: 1 | Calificacion Episodio: 3
ID Serie: S4 | Nombre Serie: Serie 4 | Duracion Episodio: 40 | Genero Episodio: Drama | Titulo Episodio: Episodio 4 | Temporada
: 1 | Calificacion Episodio: 4
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 3

De que serie le interesa conocer los episodios con cierta calificacion?
Serie 1

Ingresa la calificacion que deseas encontrar:
5
ID Serie: S1 | Nombre Serie: Serie 1 | Duracion Episodio: 50 | Genero Episodio: Drama | Titulo Episodio: Episodio 5 | Temporada
: 1 | Calificacion Episodio: 5
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 4

Con que calificacion te gustaria que te mostrara peliculas?
3
ID Pelicula: P3 | Nombre Pelicula: Ghost in the Shell 2: Innocence | Duracion Pelicula: 98 | Genero Pelicula: Ciencia Ficcion |
 Calificacion Pelicula: 3
ID Pelicula: P8 | Nombre Pelicula: Pelicula 8 | Duracion Pelicula: 150 | Genero Pelicula: Accion | Calificacion Pelicula: 3
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 5

Deseas calificar una pelicula o un episodio? (p/e)
p

Ingresa el nombre de la pelicula que deseas calificar:
Pelicula 7

Ingresa la nueva calificacion:
5

Calificacion actualizada con exito!
ID Pelicula: P7 | Nombre Pelicula: Pelicula 7 | Duracion Pelicula: 135 | Genero Pelicula: Drama | Calificacion Pelicula: 5
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 5

Deseas calificar una pelicula o un episodio? (p/e)
e

Ingresa el nombre de la serie de la cual deseas calificar un episodio:
Serie 2
Ingresa el titulo del episodio que deseas calificar:
Episodio 4

Ingresa la nueva calificacion:
3

Calificacion actualizada con exito!
ID Serie: S2 | Nombre Serie: Serie 2 | Duracion Episodio: 45 | Genero Episodio: Accion | Titulo Episodio: Episodio 4 | Temporad
a: 1 | Calificacion Episodio: 3
```

```
1. Cargar archivo de datos
2. Mostrar videos con cierta calificacion o genero
3. Mostrar episodios de una serie con cierta calificacion
4. Mostrar peliculas con cierta calificacion
5. Calificar un video
6. Salir
Selecciona una opcion: 6

Saliendo...
PS C:\Users\AlexN\poo_2do\Situacion_Problema>
```

d) Argumentación del Proyecto.

1. Cargar archivo de datos

   1. **Conteo de líneas en el archivo CSV**Se utiliza la función `countDataLinesInCSV` para determinar la cantidad de registros en el archivo CSV. Esta cantidad se utiliza para determinar el tamaño del arreglo que se creará para almacenar los datos. Si hay un error al leer el archivo, el programa muestra un mensaje de error y termina.

   2. **Creación del arreglo de películas** Se utiliza la operación `new` para crear un arreglo de objetos `Pelicula` en memoria dinámica. El tamaño del arreglo es igual al número de registros en el archivo CSV. Si no hay suficiente memoria para crear el arreglo, el programa muestra un mensaje de error y termina.

3. **Carga de datos de películas desde el archivo CSV**: Se utiliza la función `loadMoviesFromCSV` para cargar los datos de las películas desde el archivo CSV al arreglo creado en el paso anterior. Si hay un error durante la carga de datos, el programa muestra un mensaje de error, libera la memoria del arreglo de películas y termina.

4. **Creación del arreglo de episodios**: Se sigue un proceso similar al de la creación del arreglo de películas, pero en este caso para los episodios de series.

5. **Carga de datos de episodios desde el archivo CSV**: Se sigue un proceso similar al de la carga de datos de películas, pero en este caso para los episodios de series.

Se optó por la memoria dinámica para almacenar los arreglos de películas y episodios debido a que el tamaño de estos arreglos depende de la cantidad de registros en los archivos CSV, la cual no se conoce hasta que se ejecuta el programa. La memoria dinámica permite crear arreglos de un tamaño determinado en tiempo de ejecución, lo cual no es posible con la memoria estática.

Además, se optó por cargar los datos al inicio del programa (en lugar de cargarlos cada vez que se necesiten) para mejorar el rendimiento. De esta manera, una vez cargados los datos, se pueden realizar operaciones sobre ellos sin tener que leer el archivo CSV cada vez.

2. Mostrar los textos de los videos en general con una cierta calificación o de un cierto género.

1. **Verificación de la carga de datos**: Antes de proceder, el programa verifica si los datos de las películas y los episodios han sido cargados desde los archivos CSV. Si no es así, se muestra un mensaje al usuario indicando que debe cargar los archivos antes de poder usar esta opción.

2. **Selección del criterio de visualización**: Se le pregunta al usuario si desea visualizar los videos en base a su género o a su calificación. El usuario debe ingresar 'a' para género o 'b' para calificación.

3. **Visualización por género**: Si el usuario selecciona visualizar por género, se le presentan las opciones de género disponibles (Drama, Acción, Misterio). Luego, el programa recorre los arreglos de películas y episodios, y muestra aquellos cuyo género coincide con el seleccionado por el usuario.

4. **Visualización por calificación**: Si el usuario selecciona visualizar por calificación, se le pide que ingrese la calificación que desea buscar. Luego, el programa recorre los arreglos de películas y episodios, y muestra aquellos cuya calificación coincide con la ingresada por el usuario.

5. **Verificación de resultados**: Si no se encontraron videos con la calificación ingresada, se muestra un mensaje al usuario indicando que no se encontraron videos con esa calificación.

Se optó por esta solución debido a su simplicidad y eficacia. Al recorrer los arreglos de películas y episodios solo una vez, se minimiza el tiempo de ejecución. Además, al permitir al usuario seleccionar el criterio de visualización, se le proporciona un mayor control sobre la información que desea ver.

3. Mostrar los episodios de una determinada serie con una calificación determinada.

1. **Verificación de la carga de datos**: Al igual que en el caso 2, el programa verifica si los datos de las películas y los episodios han sido cargados desde los archivos CSV. Si no es así, se muestra un mensaje al usuario indicando que debe cargar los archivos antes de poder usar esta opción.

2. **Selección de la serie**: Se le pide al usuario que ingrese el nombre de la serie de la que desea ver los episodios. Se utiliza la función `getline` para leer una línea completa de entrada, ya que el nombre de la serie puede contener espacios.

3. **Selección de la calificación**: Se le pide al usuario que ingrese la calificación que desea buscar.

4. **Visualización de episodios**: El programa recorre el arreglo de episodios, y muestra aquellos cuyo nombre coincide con el nombre de la serie ingresado por el usuario y cuya calificación coincide con la calificación ingresada por el usuario.

5. **Verificación de resultados**: Si no se encontraron episodios con la calificación ingresada, se muestra un mensaje al usuario indicando que no se encontraron episodios con esa calificación.

Se optó por esta solución debido a su simplicidad y eficacia. Al recorrer el arreglo de episodios solo una vez, se minimiza el tiempo de ejecución. Además, al permitir al usuario seleccionar la serie y la calificación, se le proporciona un mayor control sobre la información que desea ver.

4. Mostrar las películas con cierta calificación.

   1. **Verificación de la carga de datos**: Al igual que en los casos anteriores, el programa verifica si los datos de las películas han sido cargados desde los archivos CSV. Si no es así, se muestra un mensaje al usuario indicando que debe cargar los archivos antes de poder usar esta opción.

   2. **Selección de la calificación**: Se le pide al usuario que ingrese la calificación que desea buscar.

   3. **Visualización de películas**: El programa recorre el arreglo de películas, y muestra aquellas cuya calificación coincide con la calificación ingresada por el usuario.

   4. **Verificación de resultados**: Si no se encontraron películas con la calificación ingresada, se muestra un mensaje al usuario indicando que no se encontraron películas con esa calificación.

   Se optó por esta solución debido a su simplicidad y eficacia. Al recorrer el arreglo de películas solo una vez, se minimiza el tiempo de ejecución. Además, al permitir al usuario seleccionar la calificación, se le proporciona un mayor control sobre la información que desea ver.

5. Calificar un video

   1. **Verificación de la carga de datos**: Al igual que en los casos anteriores, el programa verifica si los datos de las películas y los episodios han sido cargados desde los archivos CSV. Si no es así, se muestra un mensaje al usuario indicándole que debe cargar los archivos antes de poder usar esta opción.

   2. **Selección del tipo de video**: Se le pide al usuario que indique si desea calificar una película o un episodio. El usuario debe ingresar 'p' para película o 'e' para episodio.

   3. **Selección de la película o episodio**: Si el usuario selecciona calificar una película, se le pide que ingrese el nombre de la película. Si el usuario selecciona calificar un episodio, se le pide que ingrese el nombre de la serie y el título del episodio.

   4. **Ingreso de la calificación**: Se le pide al usuario que ingrese la nueva calificación.

5. **Actualización de la calificación**: El programa recorre el arreglo de películas o episodios, según corresponda, y actualiza la calificación del video seleccionado por el usuario. Si no se encuentra el video, se muestra un mensaje al usuario indicando que no se encontró el video con el nombre proporcionado.

   Se optó por esta solución debido a su simplicidad y eficacia. Al recorrer el arreglo de películas o episodios solo una vez, se minimiza el tiempo de ejecución. Además, al permitir al usuario calificar los videos, se le proporciona una forma de interactuar con el catálogo de videos.

6. Salir.

   En este caso, al seleccionar la opción de salir, se le despliega un mensaje de confirmación al usuario de que está saliendo del menú, y luego se detiene la ejecución del programa.

e) Identificación de Casos de Prueba de Errores.

➔ Archivos CSV no encontrados o inaccesibles: Si los archivos movies.csv y series.csv no se encuentran en la misma ubicación que el ejecutable o no se pueden abrir por alguna razón (por ejemplo, si están siendo utilizados por otro programa), el programa se detendrá con un mensaje de error.

➔ Formato de archivos CSV incorrecto: Si los archivos CSV no siguen el formato esperado, las funciones loadMoviesFromCSV y loadSeriesFromCSV podrían fallar. Esto podría suceder si faltan campos, si los campos están en un orden incorrecto, si los tipos de datos no son los esperados, etc.

➔ Falta de memoria: Si el sistema no tiene suficiente memoria para asignar los arreglos peliculaArray y episodioArray, el programa se detendrá con un mensaje de error.

➔ Entrada del usuario no válida: Si el usuario introduce una entrada no válida en cualquier punto (por ejemplo, una letra cuando se espera un número), esto podría causar un comportamiento inesperado. Algunas de estas situaciones se manejan en el código (por ejemplo, si el usuario elige una opción de menú no válida, se le pide que intente de nuevo), pero otras no se manejan explícitamente.

➔ Intento de operar en datos no cargados: Si el usuario intenta realizar operaciones en los datos antes de que se hayan cargado (es decir, antes de que se haya seleccionado la opción 1 del menú), el programa imprimirá un mensaje de error y no realizará la operación.

CONCLUSIÓN.

Alejandro Ignacio Vargas Cruz: El proyecto me gusto bastante, a mi parecer, este proyecto me ayudó a entender de mejor manera los conceptos relacionados con el paradigma de orientado a objetos. El plantear un modelo de clases facilitó la implementación del código, ya que nos ahorró el implementar cosas que al final hubieran sido innecesarias.De igual manera, los conceptos de herencia y polimorfismo, ayudaron a que la implementación fuera más “fácil” ya que las principales propiedades de las clases derivadas, venían de la clase base, y así, solo teníamos que implementar una pequeña cantidad de utilidades particulares para cada clase derivada. Esto, igual facilitó el poder realizar la aplicación y su menú, ya que ya teníamos “disponibles” las herramientas a utilizar, solo era cuestión de juntarlas.

Santiago Ramírez Niño: Me parece que esta modelación es bastante útil para la manipulación de grandes bases de datos que pueden estar en una plataforma de streaming, que al generar una aplicación que sólo requiere unas cuantas teclas, puede ahorrar enormes cantidades de tiempo para acceder o modificar a ciertos elementos dentro de un gran conjunto de datos. Dichos datos presentes en los archivos csv, son interpretados en el código como objetos de video gracias a los conceptos de abstracción, encapsulamiento y herencia estipulados en las clases creadas (Video es la clase base, las demás heredan de ella). Gracias al polimorfismo podemos realizar varias operaciones con distintos tipos de datos, pero que en esencia son las mismas, al sobrecargar funciones o realizar enlaces dinámicos. Los apuntadores y referencias también son fundamentales para un aprovechamiento óptimo de memoria, desde solicitar memoria dinámica en el heap hasta evitar copias de atributos innecesariamente en valores de retorno de métodos. Lo aprendido durante estas clases pueden ayudar de sobremanera a no sólo realizar códigos que nos ayuden a resolver problemas automáticamente y computacionalmente, sino a hacerlo de una manera óptima, eficiente y limpia para un adecuado manejo de recursos y de tiempos de compilación y ejecución.

Ambos consideramos que la programación orientada a objetos facilita demasiado la ejecución de diversas tareas, y que verdaderamente el paradigma de objetos nos ayuda a recrear el mundo real en una computadora, estableciendo patrones para diversos escenarios y reutilizando código para los mismos propósitos en diferentes escenarios. La reutilización del código facilita el mantenimiento y la expansión de proyectos complejos, la encapsulación garantiza la integridad y seguridad de los datos, la herencia permite crear

jerarquías de clases que fomentan la reutilización y establecer relaciones entre ellas, y el polimorfismo ofrece flexibilidad para manejar diferentes tipos de objetos en una misma interfaz. La POO es esencial para el desarrollo de aplicaciones modernas, optimizando y automatizando el proceso de programación y el resultado final.

REFERENCIAS.

Todas fueron consultadas entre el 10 y 12 de junio del 2024.

GeeksforGeeks. (2023, 16 noviembre). *C polymorphism*. GeeksforGeeks. https://www.geeksforgeeks.org/cpp-polymorphism/

GeeksforGeeks. (2024, 6 junio). *Inheritance in C*. GeeksforGeeks. https://www.geeksforgeeks.org/inheritance-in-c/

UML Class Diagram Tutorial. (s. f.). https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Virtual/pure virtual explained. (s. f.). Stack Overflow. https://stackoverflow.com/questions/1306778/virtual-pure-virtual-explained

When and why do I need to use cin.ignore() in C++? (s. f.). Stack Overflow. https://stackoverflow.com/questions/25475384/when-and-why-do-i-need-to-use-cin-ignore-in-c