



**Tecnológico  
de Monterrey**

**Ingeniería Tecnologías de información.**

**Programación Orientada a Objetos  
TC1030**

**Modelado de Servicio de Streaming**

José Antonio León Navarro A01639250

Eugenio Santisteban Zolezzi A01720932

## Contenido

Introducción	3
Diagrama de Clases UML	4
Ejemplo de ejecución.	5
Cargar Archivo de Datos	5
Mostrar los Vídeos con un cierto Género.	5
Mostrar los Vídeos con una cierta calificación.	6
Mostrar los episodios de una determinada serie con una calificación determinada.	6
Consultar una Serie con Episodios.	7
Consultar Películas con cierta Calificación.	7
Calificar un vídeo.	8
Argumentación	8
Identificación de casos que harían que el proyecto deje de funcionar.	19
Conclusión Personal.	20
Referencias.	20

## Introducción

En los últimos años, han proliferado los servicios de streaming de video bajo demanda por ejemplo Netflix, Disney, DC entre otros. Algunos de ellos se especializan por el volumen de videos que proporcionan a sus usuarios mientras que otros se han puesto el reto de mostrar solamente videos de su propia marca. Una versión limitada para apoyar a un futuro proveedor de este tipo de servicios es la que se describe a continuación:

Se quiere trabajar con dos tipos de videos: películas y series. Todo video tiene un ID, un nombre, una duración y un género (drama, acción, misterio).

Las series tienen episodios y cada episodio tiene un título y temporada a la que pertenece.

Nos interesa conocer la calificación promedio que ha recibido cada uno de los videos. Esta calificación está en escala de 1 a 5 donde 5 es la mejor calificación.

El sistema debe ser capaz de:

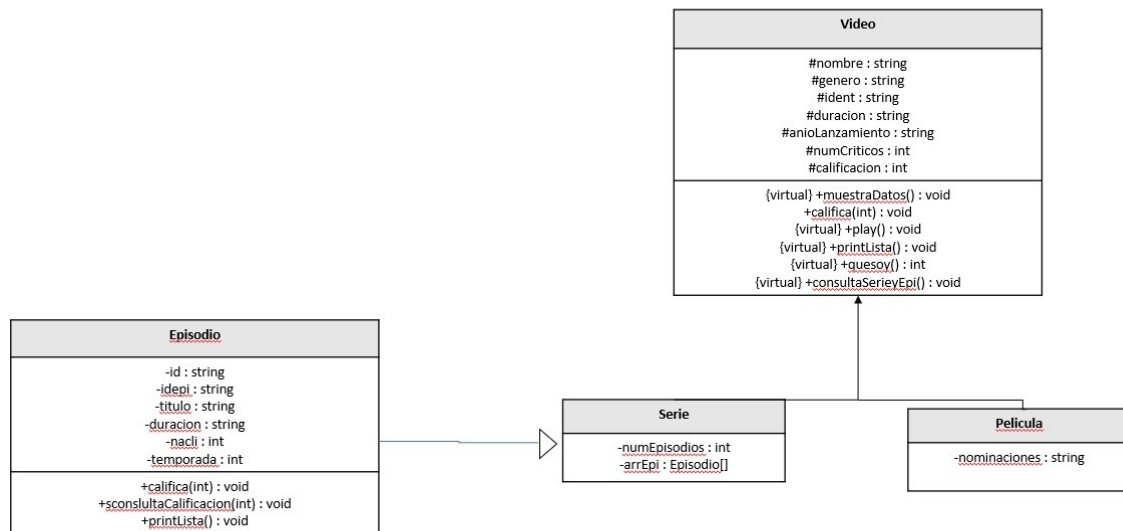
- Mostrar los videos en general con sus calificaciones
- Mostrar los episodios de una determinada serie con sus calificaciones
- Mostrar las películas con sus calificaciones

¿Cuál sería el Diagrama de clases que representaría la situación planteada?

¿Cómo emplearías los conceptos de Programación Orientada a Objetos tales como herencia, polimorfismo y sobrecarga de operadores para construir el sistema de clases que facilitaría la solución de la situación problema?

¿Podrías construir una aplicación que tome la información sobre los diferentes tipos de videos y genere reportes como: películas de un cierto género, series de un cierto género, ¿películas con su calificación?

## Diagrama de Clases UML



Para manejar la herencia heredé de vídeo a las clases derivadas serie y película, de esta forma la clase base, la cual es vídeo, le transmite a serie y película todo lo que ya tiene. Además, la herencia tiene como gran ventaja el reuso de código. Las tres clases tienen atributos y métodos en común lo que justifica la herencia.

Serie se compone de episodios por eso tienen una relación de composición. En realidad, serie se compone de un arreglo de episodios.

## Ejemplo de ejecución.

Cargar Archivo de Datos

```
File Edit Selection View Go Run Terminal Help main.cpp - Entrega_Po...
C Pelicula.cpp C Pelicula.h C Serie.h C Serie.cpp C Episodio.h C Episodio.cpp
main.cpp > leeDatosPeli()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
1
Pelicula--->1, Ad Astra, Drama, 120, 5, 2015
Pelicula--->2, The Platform, Drama, 150, 4, 2014
Pelicula--->3, Star Wars: The Rise of Skywalker, Accion, 138, 2, 2019
Pelicula--->4, The Lion King, Drama, 124, 2, 2020
Pelicula--->5, Onward, Accion, 165, 1, 2016
Pelicula--->6, Extraction, Drama, 142, 5, 2018
Pelicula--->7, Beauty and the Beast, Accion, 136, 4, 2019
Serie--->20, Game of Thrones, Drama, 02:03, 5, 9
Serie--->21, Dark, Misterio, 01:34, 4, 9
Serie--->22, Stranger Things, Accion, 02:22, 3, 8
Serie--->23, La casa de papel, Drama, 01:58, 5, 9
Serie--->24, Loki, Accion, 01:48, 4, 3
Episodio--->20, 20.1, Winter is coming, 01:02, 5, 1
Episodio--->20, 20.2, The Kings Road, 00:56, 4, 1
Episodio--->20, 20.3, Lord Snow, 00:58, 3, 1
Episodio--->20, 20.4, Cripples Bastards and Broken Thingsä, 00:56, 2, 1
Episodio--->20, 20.5, The Wolf and the Lionä, 00:55, 1, 1
Episodio--->20, 20.6, A Golden Crownä, 00:53, 5, 1
Episodio--->20, 20.7, You Win or You Dieä, 00:58, 4, 1
Episodio--->20, 20.8, The Pointy Endä, 00:59, 3, 1
Episodio--->20, 20.9, Baelorä, 00:57, 2, 1
Episodio--->21, 21.1, Secretsä, 00:51, 2, 1
Episodio--->21, 21.2, Lies, 00:44, 1, 1
Episodio--->21, 21.3, Past and Present, 00:45, 5, 1
Episodio--->21, 21.4, Double Livesä, 00:47, 4, 1
Episodio--->21, 21.5, Truthsä, 00:45, 3, 1
Episodio--->21, 21.6, Sic Mundus Creatus Estä, 00:51, 2, 1
Episodio--->21, 21.7, Crossroads, 00:52, 1, 1
Episodio--->21, 21.8, "As You Sow, so You Shall Reapä", 0, 5
Episodio--->21, 21.9, Everything Is Now, 00:55, 4, 1
Episodio--->22, 22.1, Chapter One: The Vanishing of Will Byers, 00:47, 4, 1
```

Mostar los Vídeos con un cierto Género.

```
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
2
Genero a buscar (Drama - Misterio - Accion) -->Accion
Pelicula--->3, Star Wars: The Rise of Skywalker, Accion, 138, 2, 2019
Pelicula--->5, Onward, Accion, 165, 1, 2016
Pelicula--->7, Beauty and the Beast, Accion, 136, 4, 2019
Serie--->22, Stranger Things, Accion, 02:22, 3, 8
Serie--->24, Loki, Accion, 01:48, 4, 3
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
```

## Mostrar los Vídeos con una cierta calificación.

```
C:\Users\omcor\OneDrive\Documentos\ProyectoPOO2\Debug\canciones.exe
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
3
Calificacion? 3
Pelicula--->20191580 , Star Wars: The Rise of Skywalker , Accion , 2:22 , 3
Serie--->20160150 , Stranger Things , Accion , 2:22 , 3 , 25
Episodio--->20110125 , 20110125-S01E03 , Lord Snow , 0:58 , 3 , 1
Episodio--->20110125 , 20110125-S01E08 , The Pointy Endá , 0:59 , 3 , 1
Episodio--->20110125 , 20110125-S02E03 , What Is Dead May Never Dieá , 0:53 , 3 , 2
Episodio--->20110125 , 20110125-S02E08 , The Prince of Winterfellá , 0:54 , 3 , 2
Episodio--->20110125 , 20110125-S03E03 , Walk of Punishmentá , 0:56 , 3 , 3
Episodio--->20110125 , 20110125-S03E08 , Second Sonsá , 0:56 , 3 , 3
Episodio--->20110125 , 20110125-S04E03 , Breaker of Chainsá , 0:57 , 3 , 4
Episodio--->20110125 , 20110125-S04E08 , The Mountain and the Viperá , 0:52 , 3 , 4
Episodio--->20110125 , 20110125-S05E03 , High Sparrow , 1:00 , 3 , 5
Episodio--->20110125 , 20110125-S05E08 , Hardhome , 1:01 , 3 , 5
Episodio--->20110125 , 20110125-S06E03 , Oathbreakerá , 0:52 , 3 , 6
Episodio--->20110125 , 20110125-S06E08 , No One , 0:59 , 3 , 6
Episodio--->20110125 , 20110125-S07E03 , The Queen's Justice , 1:03 , 3 , 7
Episodio--->20110125 , 20110125-S08E01 , Winterfellá , 0:54 , 3 , 8
Episodio--->20110125 , 20110125-S08E06 , The Iron Throne , 1:20 , 3 , 8
Episodio--->20170120 , 20170120-S01E05 , Truthsá , 0:45 , 3 , 1
Episodio--->20170120 , 20170120-S01E10 , Alpha and Omega , 0:57 , 3 , 1
```

## Mostrar los Episodios de una determinada serie con una calificación determinada.

```
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
4
Calificacion? 3
Serie--->20160150 , Stranger Things , Accion , 2:22 , 3 , 25
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
```

## Consultar una Serie con Episodios.

```
C:\Users\omcon\OneDrive\Documents\ProyectoPOO2\Debug\canciones.exe
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
5
Serie--->20170120 , Dark , Misterio , 1:34 , 4 , 18
Serie--->20160150 , Stranger Things , Accion , 2:22 , 3 , 25
Serie--->20170220 , La casa de papel , Drama , 1:58 , 5 , 31
Serie a buscar 20170120
Serie
Id Serie: 20170120
Serie: Dark
Genero: Misterio
Duracion: 1:34
Calificacion: 4
Numero de Temporadas: 18
Id Serie 20170120 Id Episodio: 20170120-S01E01 Titulo: Secretsá Duracion: 0:51 Calificacion: 2 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E02 Titulo: Lies Duracion: 0:44 Calificacion: 1 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E03 Titulo: Past and Present Duracion: 0:45 Calificacion: 5 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E04 Titulo: Double Livesá Duracion: 0:47 Calificacion: 4 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E05 Titulo: Truthsá Duracion: 0:45 Calificacion: 3 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E06 Titulo: Sic Mundus Creatus Está Duracion: 0:51 Calificacion: 2 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E07 Titulo: Crossroads Duracion: 0:52 Calificacion: 1 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E08 Titulo: "As You Sow Duracion: so You Shall Reapá" Calificacion: 0 Numero de Temporada: 5
Id Serie 20170120 Id Episodio: 20170120-S01E09 Titulo: Everything Is Now Duracion: 0:55 Calificacion: 4 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S01E10 Titulo: Alpha and Omega Duracion: 0:57 Calificacion: 3 Numero de Temporada: 1
Id Serie 20170120 Id Episodio: 20170120-S02E01 Titulo: Beginnings and Endings Duracion: 0:53 Calificacion: 2 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E02 Titulo: Dark Matterá Duracion: 0:54 Calificacion: 1 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E03 Titulo: Ghosts Duracion: 0:56 Calificacion: 5 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E04 Titulo: The Travelers Duracion: 1:00 Calificacion: 4 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E05 Titulo: Lost and Found Duracion: 0:56 Calificacion: 3 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E06 Titulo: An Endless Cycle Duracion: 0:54 Calificacion: 2 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E07 Titulo: The White Devil Duracion: 0:58 Calificacion: 1 Numero de Temporada: 2
Id Serie 20170120 Id Episodio: 20170120-S02E08 Titulo: Endings and Beginnings Duracion: 0:57 Calificacion: 5 Numero de Temporada: 2
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
```

## Consultar Películas con cierta Calificación.

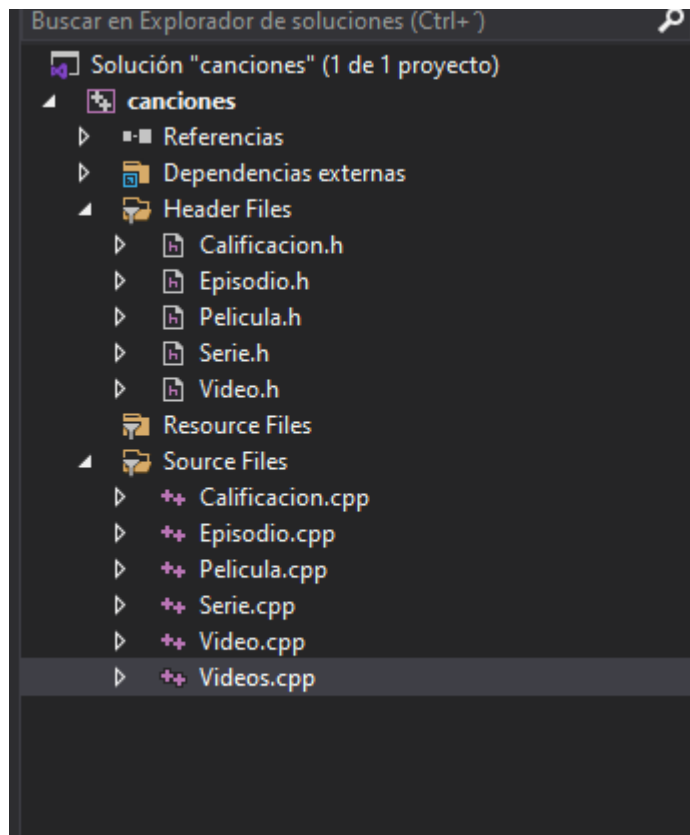
```
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
6
Calificacion? 5
Pelicula--->20192345 , Ad Astra , Drama , 2:03 , 5
Pelicula--->20200450 , Extraction , Drama , 1:56 , 5
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
```

## Calificar un Vídeo.

```
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
7
Video a calificar 20170120
  Serie
Id Serie: 20170120
Serie: Dark
Genero: Misterio
Duracion: 1:34
Calificacion: 4
Numero de Temporadas: 18
Calificacion? 5
*****Nueva Calificacion-----> 4
Serie-->20170120 , Dark , Misterio , 1:34 , 4 , 18
MENU
1. Cargar Datos
2. Consultar Videos por Genero
3. Consultar Videos por Calificacion
4. Consultar todas las series por Calificacion
5. Consultar una serie con episodios
6. Consultar peliculas con cierta calificacion
7. Calificar un video
0. Salir
```

## Argumentación

- a) Se identifican de manera correcta las clases a utilizar.





Las clases se denominan apropiadamente de acuerdo con la función que realizan, se generaron las clases que se ocupaban de acuerdo con el problema planteado y a su solución.

- b) Se emplea de manera correcta el concepto de herencia.

La herencia es uno de los paradigmas de la programación orientada a objetos. En este se tiene una clase base y una clase derivada, la cual hereda de la clase base, en este caso la clase derivada incorpora todos los miembros de la clase base además de los suyos propios.

Vídeo es la clase base y esta definida como se muestra a continuación:

```
canciones (Ámbito global)
1
2 #pragma once
3 #include <iostream>
4 #include <fstream>
5 #include <string.h>
6 #include <sstream>
7 #include <cmath>
8 #include <stdio.h>
9 #include <cstdlib>
10 // #include "Serie.h"
11 using namespace std;
12
13 class Video{
14 protected:
15     string ident, nombre, genero, duracion;
16     int calificacion;
17
18     // publico
19 public:
20     Video();
21     // constructor con parámetros
22     Video(string ideiux, string nombriux, string generiux, string duriux, int califiux);
23     // métodos de acceso para todos los atributos
24
25     string getIdentificador();
26     string getNombre();
27     string getGenero();
28     string getDuracion();
29     int getCalificacion();
30
31     // métodos de modificación para todos los atributos
32     void setIdentificador(string ideiux);
33     void setNombre(string nombriux);
34     void setGenero(string generiux);
35     void setDuracion(string duriux);
36     void setCalificacion(int califiux);
37
38     void setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux);
39     virtual void print()=0;
40     void consultaVideo();
41     virtual void printLista();
42     void califica(int valor);
43     virtual int quesoy()=0;
44     virtual void consultaSerieyEpi()=0;
45
46 };
47
```

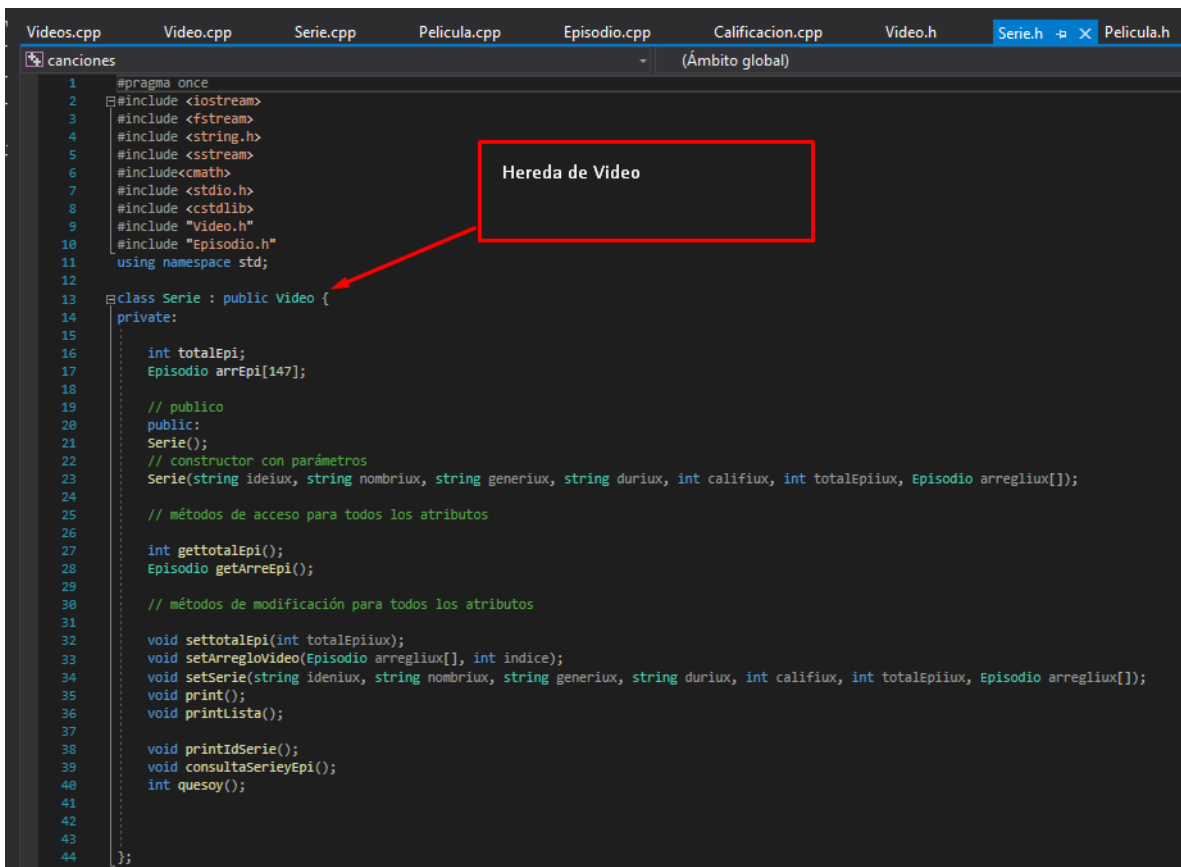
Serie y película son las clases derivadas como se muestra en las siguientes imágenes:

#### Clase derivada Película.h



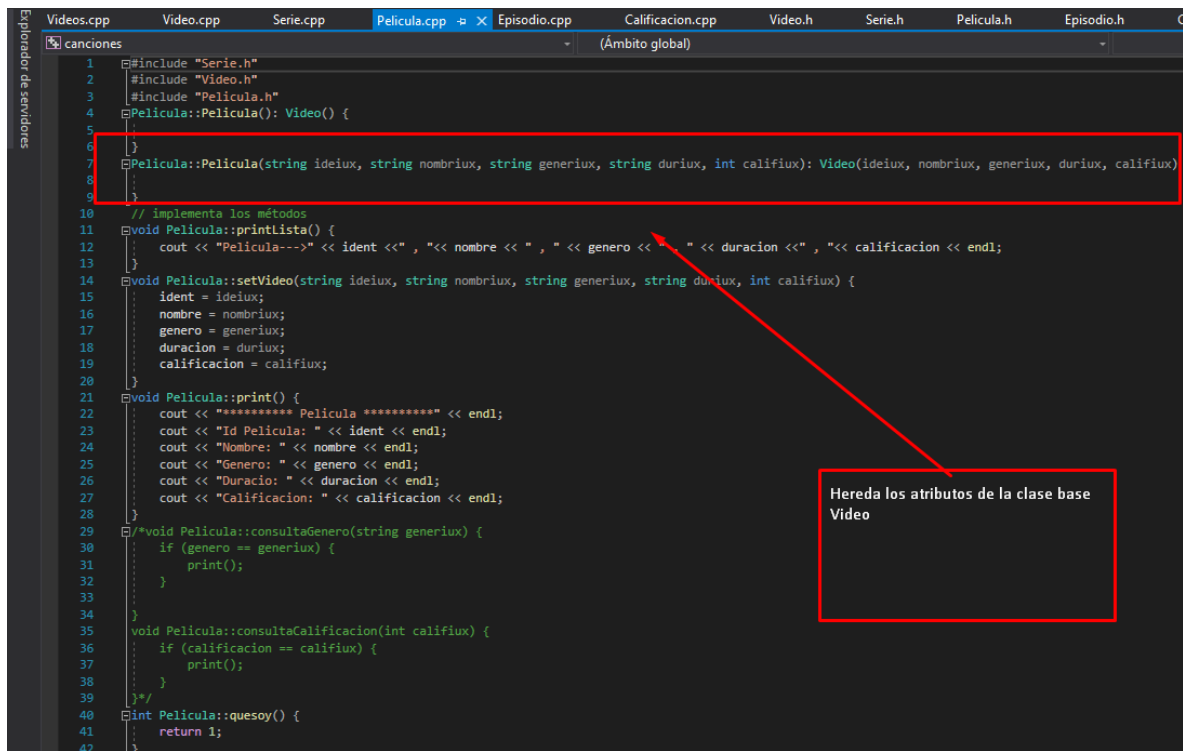
```
1 #pragma once
2 #include <iostream>
3 #include <fstream>
4 #include <string.h>
5 #include <sstream>
6 #include <cmath>
7 #include <stdio.h>
8 #include <cstdlib>
9 #include "Video.h"
10 using namespace std;
11
12 class Película : public Video {
13 public:
14     Película();
15     Película(string ideiux, string nombriux, string generiux, string duriux, int califiux);
16     void print();
17     void printLista();
18     void setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux);
19     int quesoy();
20     void consultaSerieyEpi();
21 };
```

#### Clase derivada Serie.h



```
1 #pragma once
2 #include <iostream>
3 #include <fstream>
4 #include <string.h>
5 #include <sstream>
6 #include <cmath>
7 #include <stdio.h>
8 #include <cstdlib>
9 #include "Video.h"
10 #include "Episodio.h"
11 using namespace std;
12
13 class Serie : public Video {
14 private:
15     int totalEpi;
16     Episodio arreEpi[147];
17
18     // publico
19 public:
20     Serie();
21     // constructor con parámetros
22     Serie(string ideiux, string nombriux, string generiux, string duriux, int califiux, int totalEpiux, Episodio arregliux[]);
23
24     // métodos de acceso para todos los atributos
25
26     int gettotalEpi();
27     Episodio getArreEpi();
28
29     // métodos de modificación para todos los atributos
30
31     void setttotalEpi(int totalEpiux);
32     void setArregloVideo(Episodio arregliux[], int indice);
33     void setSerie(string ideniux, string nombriux, string generiux, string duriux, int califiux, int totalEpiux, Episodio arregliux[]);
34     void print();
35     void printLista();
36
37     void printIdSerie();
38     void consultaSerieyEpi();
39     int quesoy();
40
41 };
42
43
44
45
```

## Herencia en la clase Película.cpp



```
1 #include "Serie.h"
2 #include "Video.h"
3 #include "Película.h"
4 Película::Película(): Video() {
5 }
6
7 Película::Película(string ideiux, string nombriux, string generiux, string duriux, int califiux): Video(ideiux, nombriux, generiux, duriux, califiux)
8 {
9 }
10
11 // implementa los métodos
12 void Película::printLista() {
13     cout << "Película---" << endl;
14 }
15 void Película::setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux) {
16     ident = ideiux;
17     nombre = nombriux;
18     genero = generiux;
19     duracion = duriux;
20     calificacion = califiux;
21 }
22 void Película::print() {
23     cout << "***** Película *****" << endl;
24     cout << "Id Película: " << ident << endl;
25     cout << "Nombre: " << nombre << endl;
26     cout << "Genero: " << genero << endl;
27     cout << "Duración: " << duracion << endl;
28     cout << "Calificación: " << calificacion << endl;
29 }
30 void Película::consultaGenero(string generiux) {
31     if (genero == generiux) {
32         print();
33     }
34 }
35 void Película::consultaCalificacion(int califiux) {
36     if (calificacion == califiux) {
37         print();
38     }
39 }
40 int Película::quesoy() {
41     return 1;
42 }
```

Hereda los atributos de la clase base Video

- c) Se emplea de manera correcta los modificadores de acceso  
Clase Video.h

```
Run Terminal Help Video.h - Entrega_Polimorfismo_final - Visual Studio Code
Película.cpp C Película.h C Serie.h C Serie.cpp C Episodio.h C Episodio.cpp C main.cpp C Video.cpp
Video.h > Video > califica(int)
1 #ifndef VIDEO_H
2 #define VIDEO_H
3
4 #include <iostream>
5 #include <string>
6
7 using namespace std;
8
9 class Video{
10     protected:
11         string nombre, genero, ident,duracion, anioLanzamieto;
12         int numCriticos, calificacion;
13     public:
14         Video();
15         Video(string iden, string nom, string gene, string dura, int cali, string anioLanz, int numCrit);
16
17         string getIdentificador();
18         string getNombre();
19         string getGenero();
20         string getDuracion();
21         int getCalificacion();
22         string getAnioLanzamiento();
23 }
```

Se usan atributos protegidos ya que estos serán heredados y utilizados por las clases derivadas, mientras que los métodos se mantienen públicos.

Clase Episodio.h

```
9 #include <cstdlib>
10
11 using namespace std;
12
13 class Episodio {
14     private:
15         string id;
16         string idepi;
17         string titulo;
18         string duracionE;
19         int ncali;
20         int temporada;
21
22     public:
23         Episodio();
24         Episodio(string idux, string idepiux, string tituliux, string duraiux, int ncaliux, int tempiux);
25         void print();
26         void setId(string idux);
27         void setIdepi(string idepiux);
28         void setTitulo(string tituliux);
29         void setDuracionE(string duraiux);
30         void setnCali(int ncaliux);
31         void setTemp(int tempiux);
32 }
```

Los atributos son private ya que no heredan a clases derivadas y por lo tanto no ocupan que accedan a sus atributos.

- d) Se emplea de manera correcta la sobrecarga y sobreescritura de métodos.

```
1  #pragma once
2  #include <iostream>
3  #include <fstream>
4  #include <string.h>
5  #include <sstream>
6  #include <cmath>
7  #include <stdio.h>
8  #include <cstdlib>
9  #include "Video.h"
10 using namespace std;
11
12 class Pelicula : public Video {
13
14 public:
15     Pelicula();
16     Pelicula(string ideiux, string nombriux, string generiux, string duriux, int califiux);
17     void print();
18     void printLista();
19     void setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux);
20     int quesoy();
21     void consultaSerieyEpi();
22 };

```

Estos metodos se heredan de la clase base, pero se sobrescriben en las clases derivadas.

Método printLista en la clase base Video.cpp

Video.cpp Videos.cpp Serie.cpp Pelicula.cpp Episodio.cpp Calificacion.cpp Video.h Serie.h Pelicula.h

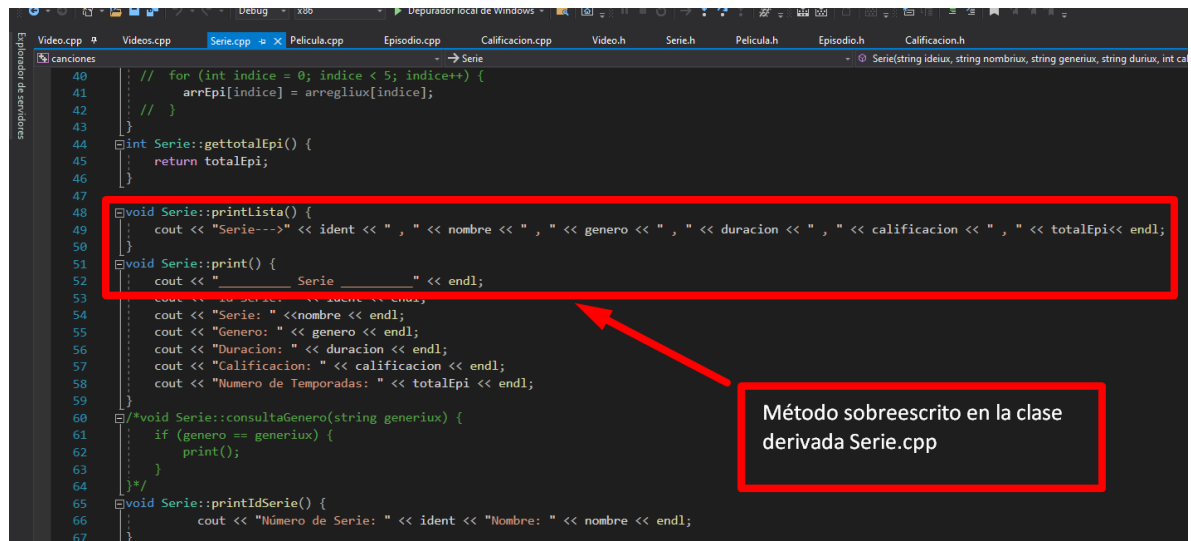
canCIONES (Ámbito global)

```
46 string Video::getNombre() {
47     return nombre;
48 }
49 string Video::getGenero() {
50     return genero;
51 }
52 string Video::getDuracion() {
53     return duracion;
54 }
55 int Video::getCalificacion() {
56     return calificacion;
57 }
58
59 void Video::consultaVideo() {
60     cout << "---- Video ----" << endl;
61     cout << "Número de Video: " << ident << endl;
62     cout << "Nombre: " << nombre << endl;
63     cout << "Genero: " << genero << endl;
64     cout << "Duracion: " << duracion << endl;
65     cout << "Calificacion: " << calificacion << endl;
66 }
67 void Video::printLista() {
68     cout << "Video --->" << ident << nombre << genero << duracion << calificacion << endl;
69 }
70
71 void Video::califica(int valor) {
72     calificacion = (calificacion + valor) / 2;
73     cout << "*****Nueva Calificacion----->" << calificacion << endl;
74 }
75

```

Método original en la clase base, Video.cpp

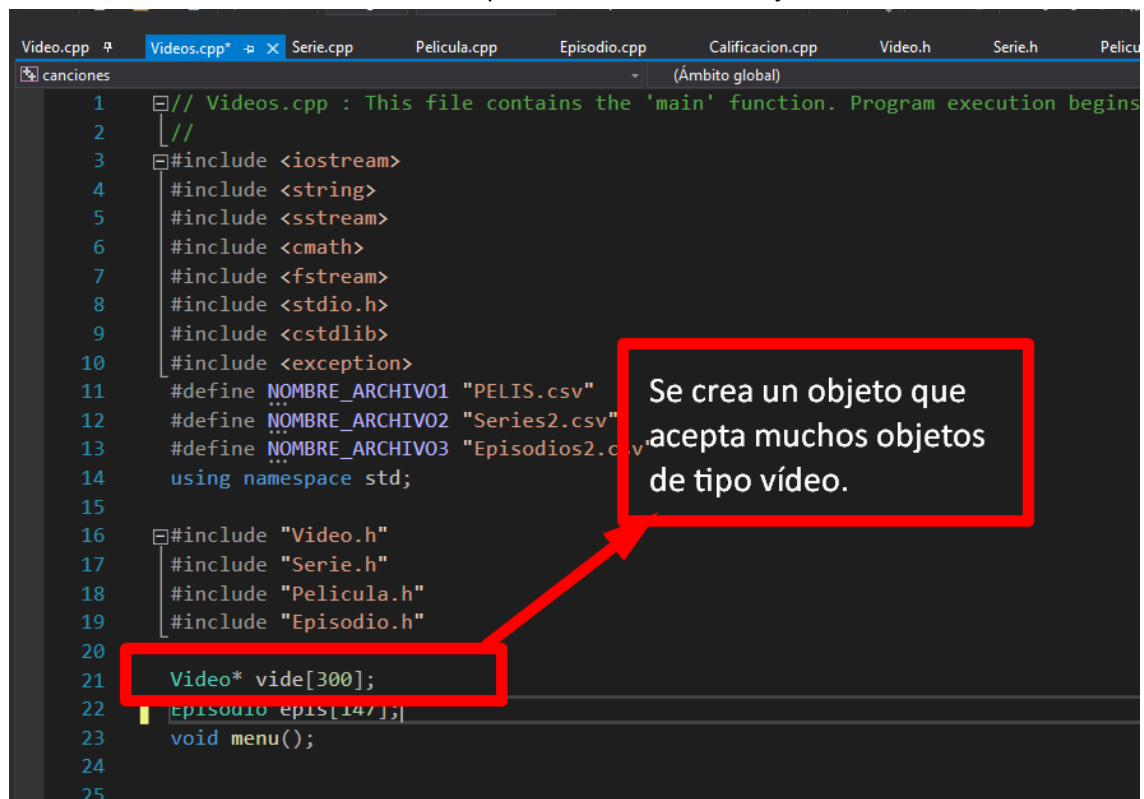
Método sobrescrito printLista en la clase serie.



```
40 // for (int indice = 0; indice < 5; indice++) {
41 //     arrEpi[indice] = arregliux[indice];
42 // }
43
44 int Serie::gettotalEpi() {
45     return totalEpi;
46 }
47
48 void Serie::printLista() {
49     cout << "Serie-->" << ident << " , " << nombre << " , " << genero << " , " << duracion << " , " << calificacion << " , " << totalEpi << endl;
50 }
51 void Serie::print() {
52     cout << "Serie: " << nombre << endl;
53     cout << "Duracion: " << duracion << endl;
54     cout << "Genero: " << genero << endl;
55     cout << "Calificacion: " << calificacion << endl;
56     cout << "Numero de Temporadas: " << totalEpi << endl;
57 }
58
59 void Serie::consultaGenero(string generiux) {
60     if (genero == generiux) {
61         print();
62     }
63 }
64
65 void Serie::printIdSerie() {
66     cout << "Número de Serie: " << ident << "Nombre: " << nombre << endl;
67 }
```

Método sobrescrito en la clase derivada Serie.cpp

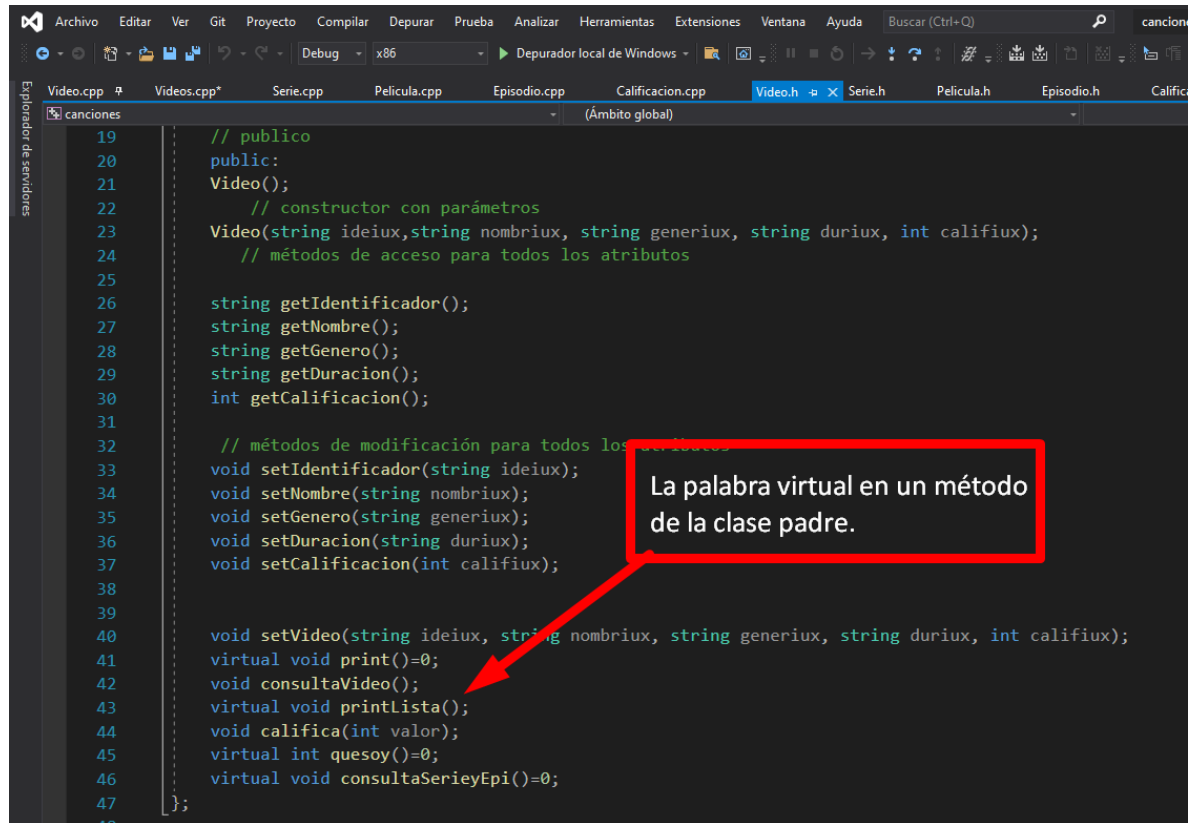
- e) Se emplea de manera correcta el concepto de polimorfismo. Polimorfismo consiste en dar distintos comportamientos a un objeto de acuerdo con la situación. Se puede observar en sobrecarga de funciones, métodos virtuales, etc. Se crea un objeto de tipo video que acepte muchos objetos de tipo video(que es el padre) y que guarda una referencia a los hijos, de tal forma que si se manda llamar a los métodos virtuales, se mande llamar al método requerido de acuerdo al objeto del cual se trate.



```
1 // Videos.cpp : This file contains the 'main' function. Program execution begins
2 //
3 #include <iostream>
4 #include <string>
5 #include <sstream>
6 #include <cmath>
7 #include <fstream>
8 #include <stdio.h>
9 #include <cstdlib>
10 #include <exception>
11 #define NOMBRE_ARCHIVO1 "PELIS.csv"
12 #define NOMBRE_ARCHIVO2 "Series2.csv"
13 #define NOMBRE_ARCHIVO3 "Episodios2.csv"
14 using namespace std;
15
16 #include "Video.h"
17 #include "Serie.h"
18 #include "Película.h"
19 #include "Episodio.h"
20
21 Video* vide[300];
22 Episodio epis[147];
23 void menu();
24
25
```

Se crea un objeto que acepta muchos objetos de tipo vídeo.

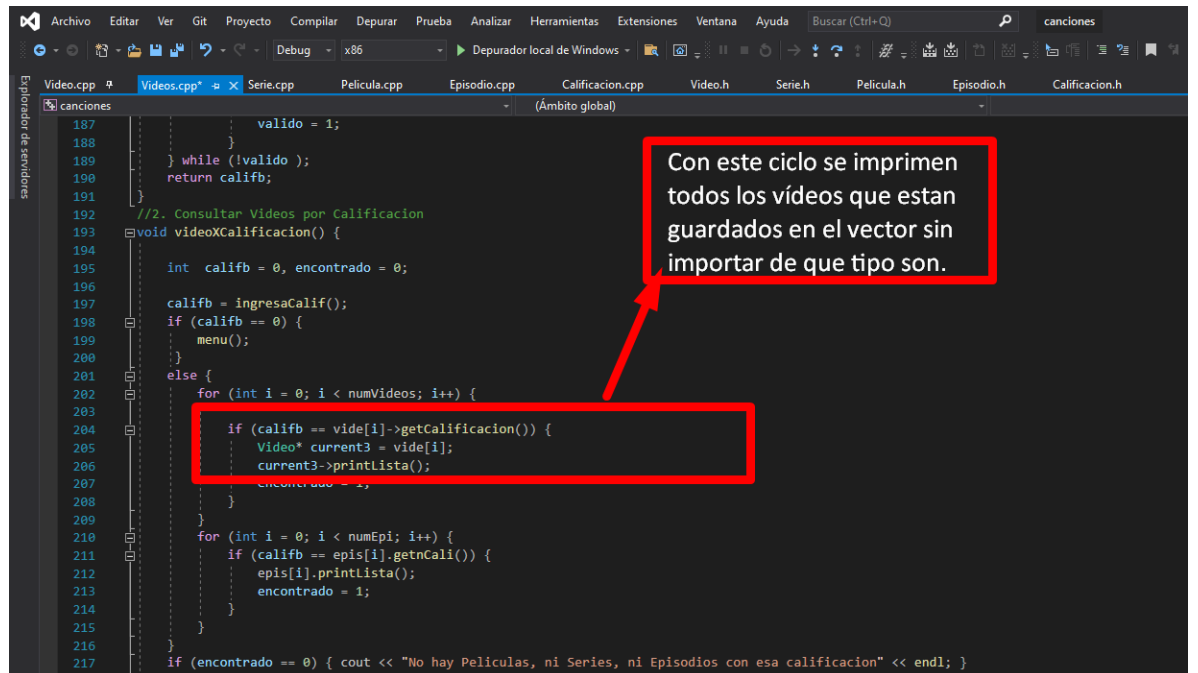
## Palabra reservada virtual en el método de la clase padre



```
19 // publico
20 public:
21 Video();
22 // constructor con parámetros
23 Video(string ideiux, string nombriux, string generiux, string duriux, int califiux);
24 // métodos de acceso para todos los atributos
25
26 string getIdentificador();
27 string getNombre();
28 string getGenero();
29 string getDuracion();
30 int getCalificacion();
31
32 // métodos de modificación para todos los atributos
33 void setIdentificador(string ideiux);
34 void setNombre(string nombriux);
35 void setGenero(string generiux);
36 void setDuracion(string duriux);
37 void setCalificacion(int califiux);
38
39 void setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux);
40 virtual void print()=0;
41 void consultaVideo();
42 virtual void printLista();
43 void califica(int valor);
44 virtual int quesoy()=0;
45 virtual void consultaSerieyEpi()=0;
46
47 };
```

La palabra virtual en un método de la clase padre.

## Las ventajas del polimorfismo se pueden notar en la siguiente imagen.

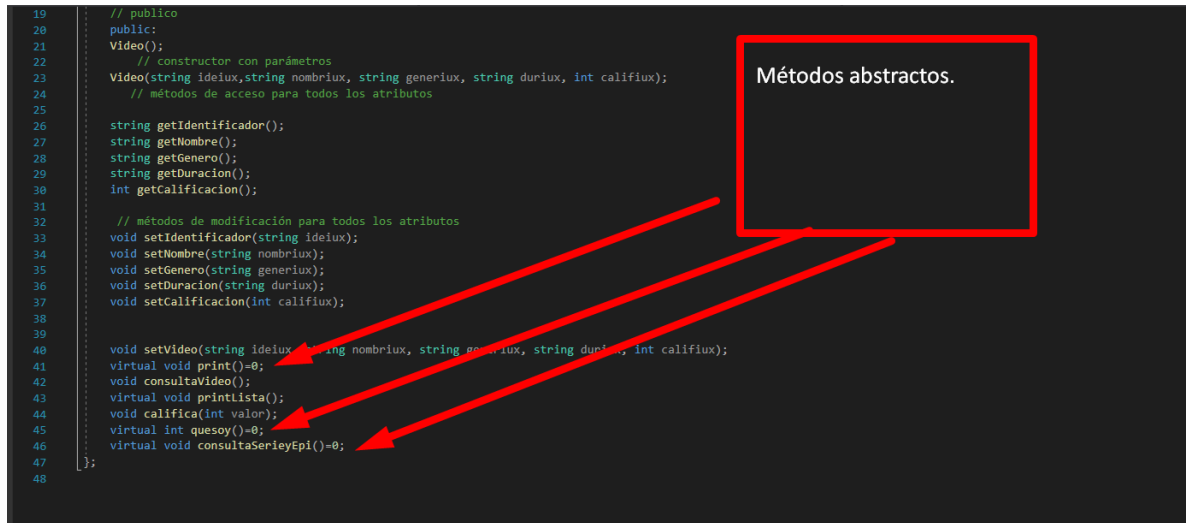


```
187 valido = 1;
188 }
189 } while (!valido);
190 return califb;
191
192 //2. Consultar Videos por Calificacion
193 void videoXCalificacion() {
194     int califb = 0, encontrado = 0;
195
196     califb = ingresaCalif();
197     if (califb == 0) {
198         menu();
199     }
200     else {
201         for (int i = 0; i < numVideos; i++) {
202             if (califb == vide[i]->getCalificacion()) {
203                 Video* current3 = vide[i];
204                 current3->printLista();
205                 encontrado = 1;
206             }
207         }
208         for (int i = 0; i < numEpi; i++) {
209             if (califb == epis[i].getnCalif()) {
210                 epis[i].printLista();
211                 encontrado = 1;
212             }
213         }
214     }
215     if (encontrado == 0) { cout << "No hay Peliculas, ni Series, ni Episodios con esa calificacion" << endl; }
216 }
```

Con este ciclo se imprimen todos los vídeos que estan guardados en el vector sin importar de que tipo son.

- f) Se emplea de manera correcta el concepto de clases abstractas.

Las clases abstractas no se pueden instanciar por lo que generalmente se usan como base para las clases hijas. En la siguiente imagen se muestra la clase abstracta Video.h y los métodos indicando que son abstractos.



```
19 // publico
20 public:
21 Video();
22 // constructor con parámetros
23 Video(string ideiux, string nombriux, string generiux, string duriux, int califiux);
24 // métodos de acceso para todos los atributos
25
26 string getIdentificador();
27 string getNombre();
28 string getGenero();
29 string getDuracion();
30 int getCalificacion();
31
32 // métodos de modificación para todos los atributos
33 void setIdentificador(string ideiux);
34 void setNombre(string nombriux);
35 void setGenero(string generiux);
36 void setDuracion(string duriux);
37 void setCalificacion(int califiux);
38
39
40 void setVideo(string ideiux, string nombriux, string generiux, string duriux, int califiux);
41 virtual void print()=0;
42 void consultaVideo();
43 virtual void printLista();
44 void califica(int valor);
45 virtual int quesoy()=0;
46 virtual void consultaSerieyEpi()=0;
47 };
48
```

Métodos abstractos.

- g) Se sobrecarga al menos un operador en el conjunto de clases propuestas.

Los operadores son símbolos (o palabras) que representan y ejecutan una operación en C++. Así como se puede sobrecargar funciones en C++ también se puede sobrecargar operadores para que tengan una funcionalidad distinta a la original o para que puedan ser implementados en tipos de datos que no los soportan de forma nativa, como las clases que implementa un desarrollador.



```
19     int ncali;
20     int temporada;
21
22
23     public:
24     Episodio();
25     Episodio(string idux, string idepiux, string tituliux, string duraiux, int ncaliux, int tempuiux);
26     void print();
27     void setId(string idux);
28     void setIdepi(string idepiux);
29     void setTitulo(string tituliux);
30     void setDuracionE(string duraiux);
31     void setnCali(int ncaliux);
32     void setTemp(int tempuiux);
33     string getId();
34     string getIdepi();
35     string getTitulo();
36     string getDuracionE();
37     int getnCali();
38     int getTemp();
39     void califica(int valor);
40     void consultaCalificacion(int ncaliux);
41     void printLista();
42     void setEpi(string idux, string idepiux, string tituliux, string duraiux, int ncaliux, int tempuiux);
43     friend std::ostream& operator<<(std::ostream&, const Episodio&);
44 };
45
```

Se sobrecarga el operador "<<".

Implementación del operador sobrecargado.

```
61     cout << "Id Episodio: " << idepi << endl;
62     cout << "Titulo: " << titulo << endl;
63     cout << "Duracion: " << duracionE << endl;
64     cout << "Calificacion: " << ncali << endl;
65     cout << "Temporada: " << temporada << endl;
66
67 void Episodio::printLista() {
68     cout << "Episodio---" << id << " , " << idepi << " , " << titulo << " , " << duracionE << " , " << ncali << " , " << temporada << endl;
69 }
70 void Episodio::setEpi(string idux, string idepiux, string tituliux, string duraiux, int ncaliux, int tempuiux) {
71     id = idux;
72     idepi = idepiux;
73     titulo = tituliux;
74     duracionE = duraiux;
75     ncali = ncaliux;
76     temporada = tempuiux;
77 }
78 void Episodio::consultaCalificacion(int ncaliux) {
79     if (ncali == ncaliux) {
80         print();
81     }
82
83
84 ostream& operator<<(ostream& salida, const Episodio& numero) {
85     salida << "Id Serie " << numero.id << " Id Episodio: " << numero.idepi << " Titulo: " << numero.titulo << " Duracion: " << numero.duracionE << " Calificacion: " << numero.ncali << endl;
86     return salida;
87 }
88 void Episodio::califica(int valor) {
89     ncali = (ncali + valor) / 2;
90     cout << "Nueva Calificacion " << ncali << endl;
91 }
92
```

Implementación del operador sobrecargado "<<".

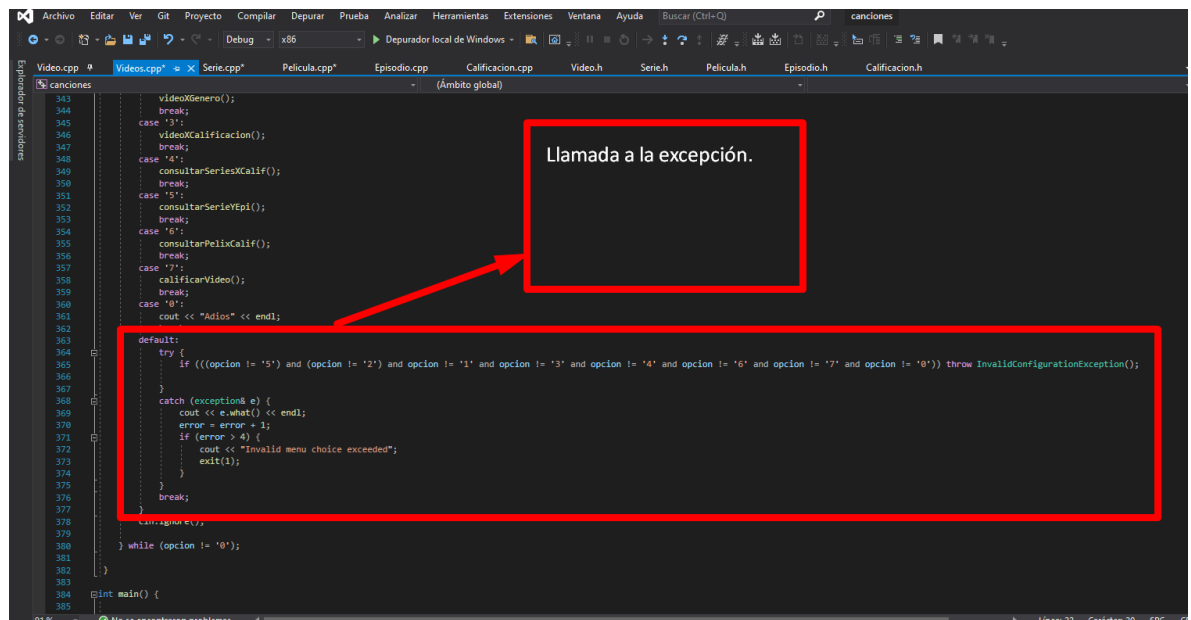
Llamada al operador sobrecargado en la clase Serie.cpp en el método consultaSerieyEpi

```
40 arrEpi[indice] = arregliux[indice];
41
42
43 int Serie::gettotalEpi() {
44     return totalEpi;
45 }
46
47 void Serie::printLista() {
48     cout << "Serie--->" << ident << " ", " << nombre << " ", " << genero << " ", " << duracion << " ", " << calificacion << " ", " << totalEpi << endl;
49 }
50 void Serie::print() {
51     cout << "Serie " << ident << endl;
52     cout << "Id Serie: " << ident << endl;
53     cout << "Serie: " << nombre << endl;
54     cout << "Genero: " << genero << endl;
55     cout << "Duracion: " << duracion << endl;
56     cout << "Calificacion: " << calificacion << endl;
57     cout << "Numero de Temporadas: " << totalEpi << endl;
58 }
59
60 void Serie::printIdSerie() {
61     cout << "Número de Serie: " << ident << "Nombre: " << nombre << endl;
62 }
63 void Serie::consultaSerieEpi() {
64     Episodio despliega;
65
66     print();
67     for (int indice = 0; indice < totalEpi; indice++) {
68         despliega = arrEpi[indice];
69         cout << despliega; //SOBRECARGA
70     }
71 }
72
73 int Serie::quesoy() {
74     return 0;
75 }
```

- h) Se utiliza de manera correcta el uso de excepciones.  
Ejemplo de aplicación de una llamada excepción. Se define la clase InvalidConfigurationException en el main.

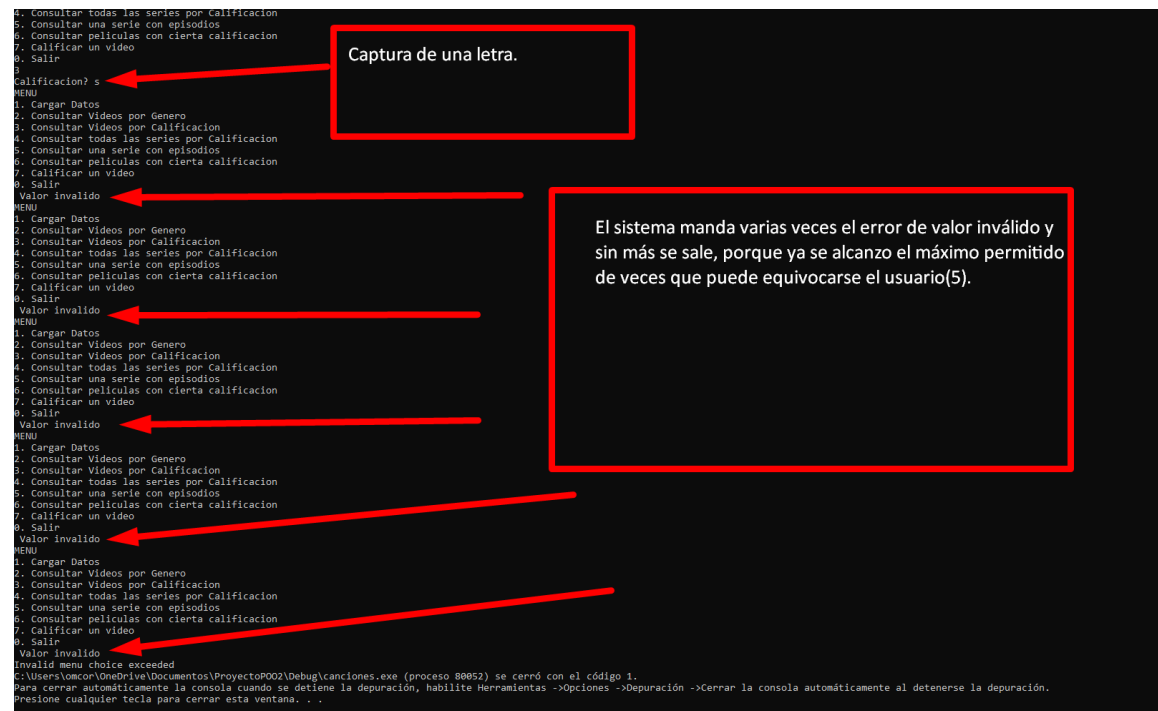
```
13 #define NOMBRE_ARCHIVO3 "Episodios2.csv"
14 using namespace std;
15
16 #include "Video.h"
17 #include "Serie.h"
18 #include "Pelicula.h"
19 #include "Episodio.h"
20
21 Video* vide[300];
22 Episodio epis[147];
23 void menu();
24
25
26 string nombre, genero, duracion, generob;
27 int numVideos=0, valor = 0, numSerie = 3, numPeli=0, numEpi = 0;
28
29 class InvalidConfigurationException : public exception {
30     virtual const char* what() const throw()
31     {
32         return " Valor invalido ";
33     }
34 } myex;
35
36 void leeDatosPeli() {
37     ifstream archivo(NOMBRE_ARCHIVO1);
38     string linea;
39     int i = 0, ncali=0;
40     char delimitador = ',';
41     // Leemos la primera línea para descartarla, pues es el encabezado
42     getline(archivo, linea);
43     // Leemos todas las líneas
44     while (getline(archivo, linea))
45     {
46         stringstream stream(linea); // Convertir la cadena a un stream
47         string id, nom, dura, genero, califi;
```

En la siguiente figura se muestra como se hace la llamada a la excepción.



## Identificación de casos que harían que el proyecto deje de funcionar.

El programa esta validado para distintas entradas que pudiera dar el usuario, excepto para cuando el sistema pide una calificación y el usuario da como entrada una letra(s), como se puede ver en la siguiente imagen:



## Conclusión Personal.

En conclusión, la programación es una rama sumamente importante y esencial en el futuro de nuestra sociedad, es una herramienta que está tomando gran importancia y es muy importante conocerla y saber aplicarla. El poder usar este conocimiento para desarrollar soluciones de la industria es algo muy interesante ya que permite trabajar con problemas reales y aplicar los conocimientos adquiridos. En general esta materia fue de gran ayuda para reforzar y comprender bien los temas relacionados a la programación orientada a objetos, como la herencia, polimorfismo, entre otros que son muy útiles para ahorrar código y recursos y hacer programas más eficientes, que es básico para el futuro de nuestra carrera y del mundo.

## Referencias.

BillWagner. (2020, March 8). Modificadores de acceso: Guía de programación de C#. Microsoft Docs.  
<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers>.

Cortijo, F. (2019). Curso de C++ Builder. Programacin Orientada a Objetos en C++.  
<https://elvex.ugr.es/decsai/builder/intro/5.html>.

Correa, C. (2020). Coding Games and Programming Challenges to Code Better. CodinGame.  
<https://www.codingame.com/playgrounds/50577/miembros-especiales-de-la-clase-en-c-practica-2/operadores-sobrecargados>.

**Link del video explicativo:** <https://youtu.be/OsDpEPROOJw>