Andrés Aguirre Rodríguez
A01284373

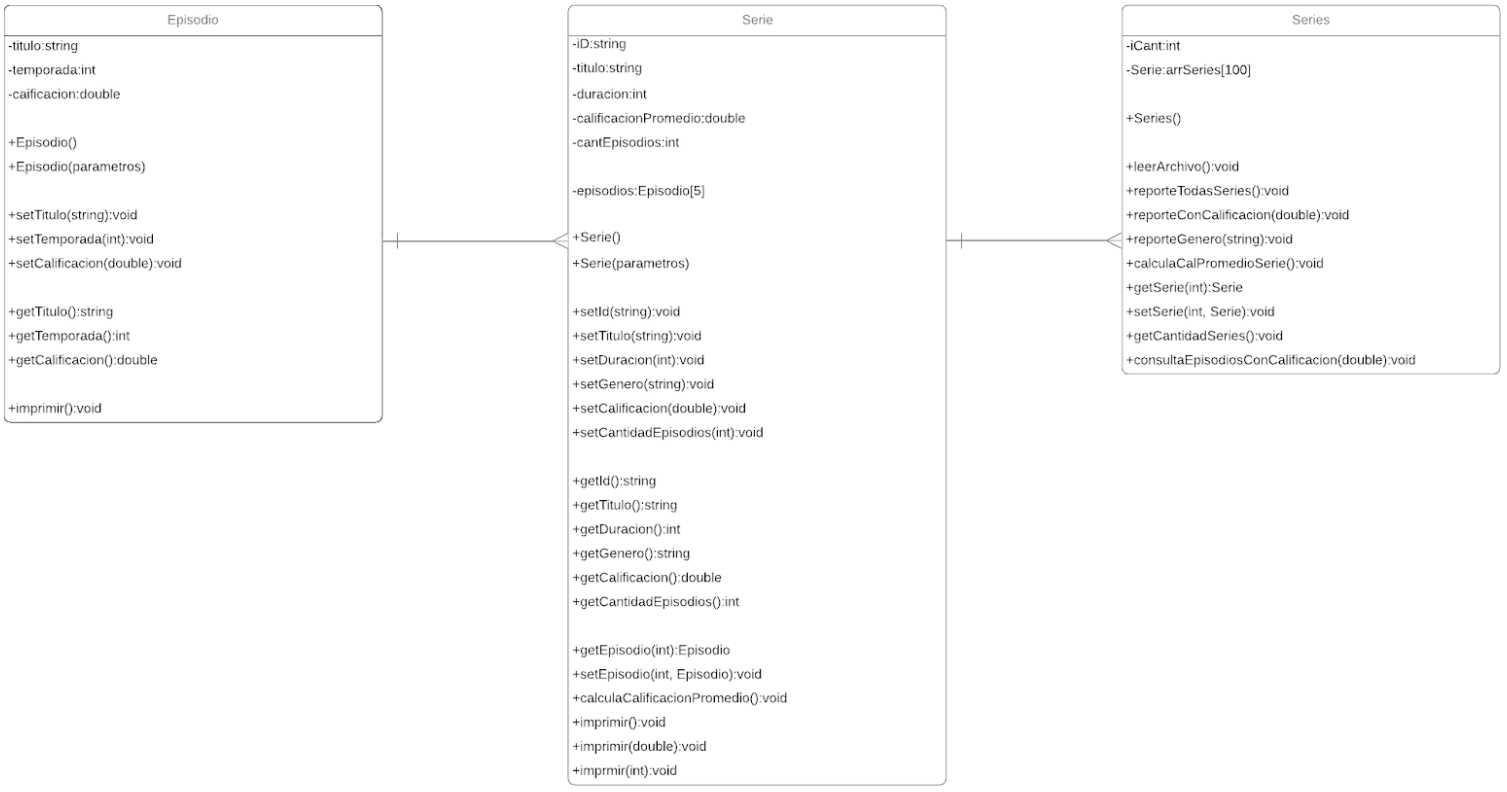
Breve Descripción de clases:

Episodio.hpp:
La clase de episodios tiene la función de usar sus métodos para crear variables en las cuales eventualmente se insertarán los elementos de cada episodio de las series solicitadas. Y aparte de poder definir las variables y sus valores, sirven de poder extraer los elementos de esas mismas variables cuando sea necesario.

Serie.hpp:
La clase de Serie tiene una función muy similar a la de la clase Episodio, sirviendo para determinar los valores de las variables que servirán para definir las series, y luego extraerlas para poder imprimirlas.

Series.hpp:
La clase de series es la que lee los documentos de clases y episodios. Utiliza las clases Episodio y Serie para definir cada una de las series y episodios las inserta en las variables para que luego puedan ser llamadas por los mismos elementos dentro de la clase, para poder segmentarlas y mostrar las que sean necesarias por caso.

Diagrama de Clases:



Archivo de Episodios:

```
101,Pilot,1,9.5
101,The Consummation,1,10
101,In a Clearing,3,9.5
101,The Reign,4,8
101,Long Live the King,1,7.2
104,Asteroid Blues,1,8.2
104,Stray dog strut,1,7.5
104,Honky tonk women,1,7.8
104,Ballad of fallen angels,1,9.2
104,The Real Folk Blues: Part 2,1,9.4
103,Everything is Fine,1,8.1
103,Janet and Michael,2,8.2
103,The Trolley Problem,2,8.7
103,The Brainy Bunch,3,7.7
103,Whenever You're Ready,4,9.6
100,The Boy in the Iceberg,1,8.1
100,The Tales of Ba Sing Se,2,9.3
100,Lake Laogai,2,9.1
100,The Crossroads of the Destiny,2,9.6
100,"Sozin's Comet Part 4: Avatar Aang",3,9.9
108,Todos Somos Cuervos,4,9.6
111,Summer Lovin,7,9.1
111,She Crazy,7,9.3
111,Express Yourself,7,9.2
111,Five Minutes,8,9.1
111,The Alliance ,8,9.5
114,That 70's Pilot,1,8.5
114,That 70's Finale,8,9
117,Chapter One: The Vanishing of Will Byers,1,9
117,Chapter One: MADMAX,2,9.3
117,"Chapter One: Suzie Do You Copy?",3,9.6
101,Me encontraste,1,9.9
101,Nos tenemos que ir,2,9.5
101,Lo que se,2,9.7
119,Good Hunting,1,8.1
120,Pilot,1,9.8
122,Pilot,1,8.3
122,The Lorelai's First Day at Yale,4,8.7
122,Bon Voyage,7,9.2
126,Pilot,1,9.5
126,Last Forever Part 2,9,9.7
126,Slap Bet,2,9.8
126,Intervention,4,9.9
131,The White Violin,1,9.5
```

```
131,The White Violin,1,9.5
131,The End of Something,2,9.2
133,The Hall of Egress,7,9.3
133,Skyhooks II,9,8.9
133,Come along with me,10,9.7
135,Made in America,6,9.5
135,Long Term Parking,5,10
135,Join the club,6,10
137,Red Moon,1,8.5
137,Prime Crew,1,9
137,Into the Abyss,1,9
137,Bent Bird,1,9.5
137,A City Upon a Hill,1,9.5
```

Archivo de Series:

```
100,Avatar: La Leyenda de Aang,22,Animación,4.9,61
101,Reign,42,Romance,4.5,78
102,Suits,43,Drama,4.25,134
103,The Good Place,22,Comedia,4.17,53
104,Cowboy Bebop,22,Animación,4.9,26
105,Breaking Bad,49,Drama,4.95,62
106,Malcolm in the middle,23,Comedia,4.7,151
107,HIMYM,22,Comedia,4.5,208
108,Club de Cuervos,42,Comedia,4.1,45
109,Peaky Blinders ,60,Drama,4.7,30
110,Steven Universe,22,Animación,4.3,160
111,Modern Family,22,Comedia,4.6,250
112,Merlin,42,Fantasía,4.2,65
113,Game of Thrones,60,Fantasía,4.7,73
114,That 70s Show,22,Comedia,4.7,200
115,The Queen's Gambit,60,Drama,5,7
116,Blacklist,45,Drama,4.8,152
117,Stranger Things,42,Ciencia ficcion,5,25
118,The boys,50,ficcion superheroes,4.7,16
119,Love Death + Robots,20,Ciencia ficcion,4.25,18
120,Full Metal Alchemis Brotherhood,23,Animación,4.9,64
121,Gossip Girl,42,Drama,4.8,121
122,Gilmore Girls,44,Drama,4.8,153
123,Dark,50,Ciencia ficcion,4.5,26
124,Kokoro Connect,25,Romance,4.9,13
125,Midhunter  ,60,Thriller,4.3,19
126,How I Met Your Mother,22,Comedia,5,208
127,One Punch Man,24,Animación,4.5,12
128,Glee,43,Drama,4.8,121
129,Game of thrones,60,Fantasy,5,73
130,Money Heist,59,Thriller,4.5,31
131,The Umbrella Academy,53,Acción,4.3,20
132,Trollhunters ,24,Fantasía ,4.9,52
133,Adventure Time,11,Fantasía,5,283
134,Erased,24,Animación,4.45,24
135,The Sopranos,50,Drama,5,86
136,Gotham,45,Ficcion ,4.2,82
137,For All Mankind,60,Drama,4.5,10
```

Casos de Prueba:

Caso 1:

```
PS C:\Users\Admin\Documents\VS Code\Avance3> cd "c:\Users\Admin\Documents\VS Code\Avance3\" ; if ($?) { g++ -std=c++17 main.cpp -o main } ; if ($?) { .\main }
1
105, Breaking Bad, Drama, 49, 0, 4.95
105, Breaking Bad, Drama, 49, 0, 4.95
ReporteSeries
105, Breaking Bad, Drama, 49, 0, 0

101, Reign, Romance, 42, 5, 8.84
Pilot, 1, 9.5
The Consummation, 1, 10
In a Clearing, 3, 9.5
The Reign, 4, 8
Long Live the King, 1, 7.2

102, Suits, Drama, 43, 0, 0

103, The Good Place, Comedia, 22, 5, 8.46
Everything is Fine, 1, 8.1
Janet and Michael, 2, 8.2
The Trolley Problem, 2, 8.7
The Brainy Bunch, 3, 7.7
Whenever You're Ready, 4, 9.6

104, Cowboy Bebop, Animaci├│n, 22, 5, 8.42
Asteroid Blues, 1, 8.2
Stray dog strut, 1, 7.5
Honky tonk women, 1, 7.8
Ballad of fallen angels, 1, 9.2
The Real Folk Blues: Part 2, 1, 9.4

105, Breaking Bad, Drama, 49, 0, 0

106, Malcolm in the middle, Comedia, 23, 0, 0

107, HIMYM, Comedia, 22, 0, 0

108, Club de Cuervos, Comedia, 42, 1, 9.6
Todos Somos Cuervos, 4, 9.6

109, Peaky Blinders , Drama, 60, 0, 0

110, Steven Universe, Animaci├│n, 22, 0, 0

111, Modern Family, Comedia, 22, 5, 9.24
Summer Lovin, 7, 9.1
She Crazy, 7, 9.3
Express Yourself, 7, 9.2
Five Minutes, 8, 9.1
The Alliance , 8, 9.5

112, Merlin, Fantas├¡a, 42, 0, 0

113, Game of Thrones, Fantas├¡a, 60, 0, 0
```

Caso 2:

```
PS C:\Users\Admin\Documents\VS Code\Avance3> cd "c:\Users\Admin\Documents\VS Code\Avance3\" ; if ($?) { g++ -std=c++17 main.cpp -o main } ; if ($?) { .\main }
2
38
ReporteEpisodiosConCalificacion:9.5
101, Reign, Romance, 42, 5, 4.5
Pilot, 1, 9.5
In a Clearing, 3, 9.5

111, Modern Family, Comedia, 22, 5, 4.6
The Alliance , 8, 9.5

126, How I Met Your Mother, Comedia, 22, 4, 5
Pilot, 1, 9.5

131, The Umbrella Academy, Acci├│n, 53, 2, 4.3
The White Violin, 1, 9.5

135, The Sopranos, Drama, 50, 3, 5
Made in America, 6, 9.5

137, For All Mankind, Drama, 60, 5, 4.5
Bent Bird, 1, 9.5
A City Upon a Hill, 1, 9.5
```

Caso 3:

```
PS C:\Users\Admin\Documents\VS Code\Avance3> cd "c:\Users\Admin\Documents\VS Code\Avance3\" ; if ($?) { g++ -std=c++17 main.cpp -o main } ; if ($?) { .\main }
3
ReporteSeriesConCalificacion:4.5
101, Reign, Romance, 42, 5, 4.5
Pilot, 1, 9.5
The Consummation, 1, 10
In a Clearing, 3, 9.5
The Reign, 4, 8
Long Live the King, 1, 7.2

107, HIMYM, Comedia, 22, 0, 4.5

123, Dark, Ciencia ficcion, 50, 0, 4.5

127, One Punch Man, Animaci├│n, 24, 0, 4.5

130, Money Heist, Thriller, 59, 0, 4.5

137, For All Mankind, Drama, 60, 5, 4.5
Red Moon, 1, 8.5
Prime Crew, 1, 9
Into the Abyss, 1, 9
Bent Bird, 1, 9.5
A City Upon a Hill, 1, 9.5
```

Caso 4:

```
PS C:\Users\Admin\Documents\VS Code\Avance3> cd "c:\Users\Admin\Documents\VS Code\Avance3\" ; if ($?) { g++ -std=c++17 main.cpp -o main } ; if ($?) { .\main }
4
ReporteSeriesConCalificacion:0
102, Suits, Drama, 43, 0, 0

105, Breaking Bad, Drama, 49, 0, 0

106, Malcolm in the middle, Comedia, 23, 0, 0

107, HIMYM, Comedia, 22, 0, 0

109, Peaky Blinders , Drama, 60, 0, 0

110, Steven Universe, Animaci├│n, 22, 0, 0

112, Merlin, Fantas├¡a, 42, 0, 0

113, Game of Thrones, Fantas├¡a, 60, 0, 0

115, The Queen's Gambit, Drama, 60, 0, 0

116, Blacklist, Drama, 45, 0, 0

118, The boys, ficcion superheroes, 50, 0, 0

121, Gossip Girl, Drama, 42, 0, 0

123, Dark, Ciencia ficcion, 50, 0, 0

124, Kokoro Connect, Romance, 25, 0, 0

125, Midhunter  , Thriller, 60, 0, 0

127, One Punch Man, Animaci├│n, 24, 0, 0

128, Glee, Drama, 43, 0, 0

129, Game of thrones, Fantasy, 60, 0, 0

130, Money Heist, Thriller, 59, 0, 0

132, Trollhunters , Fantas├¡a , 24, 0, 0

134, Erased, Animaci├│n, 24, 0, 0

136, Gotham, Ficcion , 45, 0, 0
```

Caso 5:

```
PS C:\Users\Admin\Documents\VS Code\Avance3> cd "c:\Users\Admin\Documents\VS Code\Avance3\" ; if ($?) { g++ -std=c++17 main.cpp -o main } ; if ($?) { .\main }
5
ReporteGenero:Ciencia ficcion
117, Stranger Things, Ciencia ficcion, 42, 3, 5
Chapter One: The Vanishing of Will Byers, 1, 9
Chapter One: MADMAX, 2, 9.3
"Chapter One: Suzie Do You Copy?", 3, 9.6

119, Love Death + Robots, Ciencia ficcion, 20, 1, 4.25
Good Hunting, 1, 8.1

123, Dark, Ciencia ficcion, 50, 0, 4.5
```

Todas las clases:

Clase Episodio.hpp:

```cpp
#include <iostream>
#include <stdio.h>
#include <string>
#pragma once
using namespace std;

// Clase Episodio
class Episodio{
    // público:
    public:
        // Constructor de omisión
        Episodio();

        // Constructor con parámetros
        Episodio(string _titulo, int _temporada, double _calificacion);

        // Métodos de modificación
        void setTitulo(string);
        void setTemporada(int);
        void setCalificacion(double);

        // Métodos de acceso
        string getTitulo();
        int getTemporada();
        double getCalificacion();

        // Método de impresión
        void imprimir();


    // privado:
    private:
        // Atributo string TITULO
        string titulo;
        // Atributo int TEMPORADA
        int temporada;
        // Atributo double CALIFICACIÓN
        double calificacion;
};

// Constructor de omisión
Episodio::Episodio(){
    titulo = "";
    temporada = 0;
    calificacion = 0.0;
}

```

```cpp
// Constructor con parámetros
Episodio::Episodio(string _titulo, int _temporada, double _calificacion){
    titulo = _titulo;
    temporada = _temporada;
    calificacion = _calificacion;
}

// Métodos de modificación
void Episodio::setTitulo(string _titulo){
    titulo = _titulo;
}

void Episodio::setTemporada(int _temporada){
    temporada = _temporada;
}

void Episodio::setCalificacion(double _calificacion){
    calificacion = _calificacion;
}

// Métodos de acceso
string Episodio::getTitulo(){
    return titulo;
}

int Episodio::getTemporada(){
    return temporada;
}

double Episodio::getCalificacion(){
    return calificacion;
}

// Método de impresión
void Episodio::imprimir(){
    cout << titulo << ", " << temporada << ", " << calificacion << endl;

}
```

Clase Serie.hpp:

```cpp
#include <iostream>
#include <stdio.h>
#include <string>
#include "Episodio.hpp"
#pragma once
using namespace std;

// Clase Serie
class Serie{
    // público
    public:
        // Constructor de omisión
        Serie();

        // Constructor con parámetros incluyendo el arreglo // --> NUEVO
        Serie(string _id, string _titulo, int _duracion, string _genero, double _calificacion, int _cantidad, Episodio _Episodios[]);

        // Métodos de modificación
        void setId(string);
        void setTitulo(string);
        void setDuracion(int);
        void setGenero(string);
        void setCalificacion(double);
        void setCantidadEpisodios(int);

        // Métodos de acceso
        string getId();
        string getTitulo();
        int getDuracion();
        string getGenero();
        double getCalificacion();
        int getCantidadEpisodios();

        // Método de impresión
        void imprimir();

        // Imprime la informacion de la serie pero solo los episodios que tienen cierta calificacion
        // se imprime la informacion de la serie si existe al menos un episodio con esa calificacion
        void imprimir(double _calificacion);

        // Imprime la informacion de la serie pero solo los episodios que son de la _temporada
        // se imprime la informacion de la serie si existe al menos un episodio de esa temporada
        void imprimir(int _temporada);

        // Métodos NUEVOS ---------

        // Método getEpisodio
        Episodio getEpisodio(int);

        // Método CalificaciónPromedio
        void calculaCalificacionPromedio();

        // Método setEpisodio --> Actualiza la posición iE del aarrEpisodios[], recibe
        // un objeto de la clase Episodio. Validar la posición
        void setEpisodio(int, Episodio);


    // privado:
    private:
        // Atributos string
        string iD, titulo, genero;
        // Atributos int
        int duracion, cantEpisodios;
        //Atributos double
        double calificacionPromedio;
        //
        Episodio episodios[5];
};

//---------------------- IMPLEMENTACIÓN ---------------------//

// Constructor de omisión
Serie::Serie(){
    iD = "";
    titulo = "";
    duracion = 0;
    genero = "";
    calificacionPromedio = 0;
    cantEpisodios = 0;
}

// Constructor con parámetros
Serie::Serie(string _iD, string _titulo, int _duracion, string _genero, double _calificacionP, int _cantidadE, Episodio _episodios[]){
    iD = _iD;
    titulo = _titulo;
    duracion = _duracion;
    genero = _genero;
    calificacionPromedio = _calificacionP;
    cantEpisodios = _cantidadE;
}
```

```cpp
// Métodos de modificación

void Serie::setId(string id){
    iD = id;
}

void Serie::setTitulo(string tittle){
    titulo = tittle;
}

void Serie::setDuracion(int time){
    duracion = time;
}

void Serie::setGenero(string gender){
    genero = gender;
}

void Serie::setCalificacion(double note){
    calificacionPromedio = note;
}

void Serie::setCantidadEpisodios(int cant){
    cantEpisodios = cant;
}

// Métodos de acceso
string Serie::getId(){
    return iD;
}

string Serie::getTitulo(){
    return titulo;
}

int Serie::getDuracion(){
    return duracion;
}

string Serie::getGenero(){
    return genero;
}

double Serie::getCalificacion(){
    return calificacionPromedio;
}
```

```cpp

int Serie::getCantidadEpisodios(){
    return cantEpisodios;
}

// ---------------- COMPOSICIÓN ---------------------- //

Episodio Serie::getEpisodio(int _numEpisodios){
    if (_numEpisodios >= 0 && _numEpisodios <= 4){
        return episodios[_numEpisodios];
    }
    else {
        return episodios[0];
    }
}

void Serie::setEpisodio(int _cantidadE, Episodio _episodio){
    // Verificar que _iNum está entre 1 y 4
    // Si es así, episodios[_numEpisodios]=_episodio
    if (_cantidadE >= 0 && _cantidadE < 5){
        episodios[_cantidadE]=_episodio;
    }
}


// Método CalificaciónPromedio
void Serie::calculaCalificacionPromedio(){
    double acumulador = 0.0;

    for (int x = 0; x < cantEpisodios; x++){
        acumulador = acumulador + episodios[x].getCalificacion();
    }
    if (cantEpisodios > 0){
        calificacionPromedio = acumulador / cantEpisodios;

    }
    else{
        calificacionPromedio = 0;
    }
}

```

```cpp
// Método de impresión
void Serie::imprimir(){
    cout << iD << ", " << titulo << ", " << genero << ", " << duracion << ", " << cantEpisodios << ", " << calificacionPromedio << endl;


    for (int iE = 0;iE < cantEpisodios; iE++)
        episodios[iE].imprimir();
}

void Serie::imprimir(double _calificacion){
    int primero = 1;
    double calificacion;

    for (int iE = 0; iE < cantEpisodios; iE++){
        calificacion = episodios[iE].getCalificacion();
        if (_calificacion == calificacion){
            if (primero == 1){
                cout << iD << ", " << titulo << ", " << genero << ", " << duracion << ", " << cantEpisodios << ", " << calificacionPromedio << endl;
                primero = 0;
            }
            episodios[iE].imprimir();
        }
    }
    if (cantEpisodios > 0 && primero == 0){
        cout << endl;
    }

}

void Serie::imprimir(int _temporada){
    int primero = 1;
    int temporada;

    for (int iE = 0; iE < cantEpisodios; iE++){
        temporada = episodios[iE].getCalificacion();
        if (_temporada == temporada){
            if (primero == 1){
                cout << iD << ", " << titulo << ", " << genero << ", " << duracion << ", " << cantEpisodios << ", " << calificacionPromedio << endl;
                primero = 0;
            }
            episodios[iE].imprimir();
        }
    }
    if (cantEpisodios > 0 && primero == 0){
        cout << endl;
    }
}
```

Clase Series.hpp:

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <stdio.h>
#include <string>
#include "Serie.hpp"
#include "Episodio.hpp"
using namespace std;

class Series{


    public:
        Series();

        // Constructor que inicializa iCant = 0
        Series(int iCant);

        // Lee las Series desde un archivo cvs y las carga en el arreglo y lee el archivo de Episodios
        // arrSeries[100];
        void leerArchivo();

        // Reporte de todas las Series -
        // al final del mismo muestra la calificacion promedio de todas las Series (ver casos de prueba)
        void reporteTodasSeries();

        // Reporte de Series que tienen cierta calificacion -
        // Despliega en pantalla todas las series del arreglo que tienen la calificacion recibida en el parametro de entrada
        // Si la serie tiene episodios los despliega(si los tiene)
        void reporteConCalificacion(double);

        // Reporte de Series que tienen cierto genero -
        // Despliega en pantalla todas las series que tienen el genero recibido como parametro de entrada
        // con todos sus episodios(si los tiene)
        void reporteGenero(string);

        // Calcula la calificacion promedio de todas las series en base a sus episodios y despliega su titulo y su calificacion promedio
        // Si la serie no tiene episodios, se le asigna 0 como calificacion
        void calculaCalPromedioSerie();

        // Retorna el objeto Serie que esta en la posicion iS del arreglo de series
        // que recibio como parametro de entrada, si no existe retorna la serie 0 de arreglo
        Serie getSerie(int iS);

        // Actualiza la serie que esta en la posicion iSerie del arreglo de series, validar que existe
        // si no existe NO actualiza el arreglo
        void setSerie(int iSerie, Serie s);
```

```cpp
        // Retorna la cantidad de series que tenemos en existencia - iCant
        int getCantidadSeries();

        void setCantidadSeries(int);

        // Desplegar todas las series con los episodios que tienen calificacion especificada en el parametro de entrada
        // si la serie no tiene episodios con esa calificacion no despliega nada de esa serie
        void consultaEpisodiosConCalificacion(double dCal);

        void consultarTemporadaXSerie(int temporada);

    private:
        //Arreglo de objetos de la clase Serie
        Serie arrSeries[100];
        int iCant; //cantidad de Series dadas de alta
};

Series::Series(){
    iCant = 0;

}

```

```cpp
void Series::leerArchivo(){
    string sId, sTitulo, sGenero, linea, dato;
    int iR;

    fstream lectura;

    lectura.open("Series.csv", ios::in);
    iR = 0;
    int renglon1 = 1;
    while ( getline(lectura, linea))
    {
        stringstream registro(linea);

        int columna = 0;
        while(getline(registro, dato, ','))
        {
            switch (columna++)
            {
                case 0: // iD
                    arrSeries[iR].setId(dato);
                break;
                case 1: // titulo
                    arrSeries[iR].setTitulo(dato);
                break;
                case 2: // duracion
                    arrSeries[iR].setDuracion(stoi(dato));
                break;
                case 3: // genero
                    arrSeries[iR].setGenero(dato);
                break;
                case 4: // calificacion promedio
                    arrSeries[iR].setCalificacion(stod(dato));
                break;
                case 5: //cant episodios
                    arrSeries[iR].setCantidadEpisodios(0);
                break;
            }
        }
            iR++;
    }
    iCant = iR;
    lectura.close();

```

```
114     Episodio ep;
115     lectura.open("Episodios.txt", ios::in);
116     int cantEp, iS;
117     renglon1 = 1;
118     while (getline(lectura, linea))
119     {
120         stringstream registro(linea);
121         int columna = 0;
122 
123         while(getline(registro, dato, ','))
124         {
125             switch (columna++)
126             {
127                 case 0: // iD
128                     iS = stoi(dato)-100;
129                 break;
130                 case 1: // titulo
131                    ep.setTitulo(dato);
132                 break;
133                 case 2: // temporada
134                     ep.setTemporada(stoi(dato));
135                 break;
136                 case 3: // calificacion
137                     ep.setCalificacion(stod(dato));
138                 break;
139 
140             }
141         }
142 
143         cantEp = arrSeries[iS].getCantidadEpisodios();
144         if (cantEp < 5){
145             arrSeries[iS].setEpisodio(cantEp, ep);
146             arrSeries[iS].setCantidadEpisodios(cantEp + 1);
147         }
148     }
149     lectura.close();
150 }
151 
```

```
152  void Series::reporteTodasSeries(){
153      cout << "ReporteSeries" << endl;
154      double dPromedio;
155      dPromedio = 0;
156
157
158      for(int iR = 0, iCont = 1; iR < iCant; iR++){
159          arrSeries[iR].imprimir();
160          cout << endl;
161          dPromedio = dPromedio + arrSeries[iR].getCalificacion();
162      }
163
164      if (iCant > 0)
165      {
166          cout << "Promedio Series :" << dPromedio / iCant << endl;
167      }
168  }
169
170  void Series::reporteConCalificacion(double calificacion){
171      cout << "ReporteSeriesConCalificacion:" << calificacion << endl;
172      for (int iR = 0, iCont = 1; iR < iCant; iR++){
173          if (arrSeries[iR].getCalificacion() == calificacion){
174              arrSeries[iR].imprimir();
175              cout << endl;
176          }
177
178      }
179  }
180
181  void Series::reporteGenero(string genero){
182      cout << "ReporteGenero:Ciencia ficcion" << endl;
183      for (int iR = 0, iCont = 1; iR < iCant; iR++){
184          if (arrSeries[iR].getGenero() == genero){
185              arrSeries[iR].imprimir();
186              cout << endl;
187          }
188
189      }
190  }
191
192  void Series::calculaCalPromedioSerie(){
193      for (int iR = 0, iCont = 1; iR < iCant; iR++){
194          arrSeries[iR].calculaCalificacionPromedio();
195      }
196  }
197
```

```
198  Serie Series::getSerie(int iS){
199      if (iS >= 0 && iS < iCant)
200      {
201          return arrSeries[iS];
202      }
203      else
204      {
205          return arrSeries[0];
206      }
207
208  }
209
210  void Series::setSerie(int iSerie, Serie s){
211      if (iSerie >= 0 && iSerie < iCant)
212      {
213          arrSeries[iSerie] = s;
214      }
215  }
216
217  int Series::getCantidadSeries(){
218      return iCant;
219  }
220
221  void Series::setCantidadSeries(int _iCant){
222      iCant = _iCant;
223  }
224
225
226  void Series::consultaEpisodiosConCalificacion(double dCal){
227      cout << "ReporteEpisodiosConCalificacion:9.5" << endl;
228      for (int iS = 0; iS < iCant; iS++){
229          arrSeries[iS].imprimir(dCal);
230      }
231
232  }
```

Main.cpp:

```cpp
#include <iostream>
#include <stdio.h>
#include <string>
using namespace std;

// NOTA IMPORTANTE - Añade los include que se requieran dependiendo de tu avance
#include "Serie.hpp"
#include "Episodio.hpp"
#include "Series.hpp"
```

```cpp
int main() {

    Series negocio;

    int iOpcion;

    cin >> iOpcion;

    // leer todas las series desde el archivo -
    negocio.leerArchivo();

    switch (iOpcion) {
        case 1:
            negocio.setSerie(0,  negocio.getSerie(5));
            negocio.getSerie(5).imprimir();
            negocio.getSerie(0).imprimir();
            negocio.calculaCalPromedioSerie(); //NO DESPLIEGA SOLO ACTUALIZA
            negocio.reporteTodasSeries();

            break;
        case 2:
            // Retorna la cantiadad de series que tenemos en existencia
            cout << negocio.getCantidadSeries() << endl;
            negocio.consultaEpisodiosConCalificacion(9.5);
            break;
        case 3:
            negocio.reporteConCalificacion(4.5);
            break;

        case 4:
            // Calcula la calificación promedio de todas las serie en base a sus episodios y despliega su titulo y su calificación promedio
               negocio.calculaCalPromedioSerie();
               negocio.reporteConCalificacion(0);
               //negocio.reporteTodasSeries();
            break;
        case 5:
            // Reporte de Series que tienen cierto genero -
               negocio.reporteGenero("Ciencia ficcion");
            break;

    }


    return 0;
}
```

Evidencia de casos de Vocareum:

Avance 1:

Avance 2:

Avance 3: