Tecnológico de Monterrey, Campus Sinaloa

Escuela de Ingeniería y Ciencias

Proyecto Integrador

Por:

Juan Pablo Bernal Lafarga, A01742342
Rogelio Lizarraga, A01742161
Julián Rojas Soto, A01740757

Profesor
Ricardo Rafael Quintero Meza

Programación orientada a objetos

Grupo
301

Culiacán, Sinaloa a 18 de junio de 2022

Modelado de servicio de streaming

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.

El streaming es un tipo de tecnología multimedia que envía contenidos de vídeo y audio a su dispositivo conectado a Internet. Esto le permite acceder a contenidos (TV, películas, música, podcast) en cualquier momento que lo desee.

El código contará con las siguientes características:

-Mostrar los videos(los videos tiene un ID, un nombre, una duración y un género (drama, acción, misterio) en general con sus calificaciones.

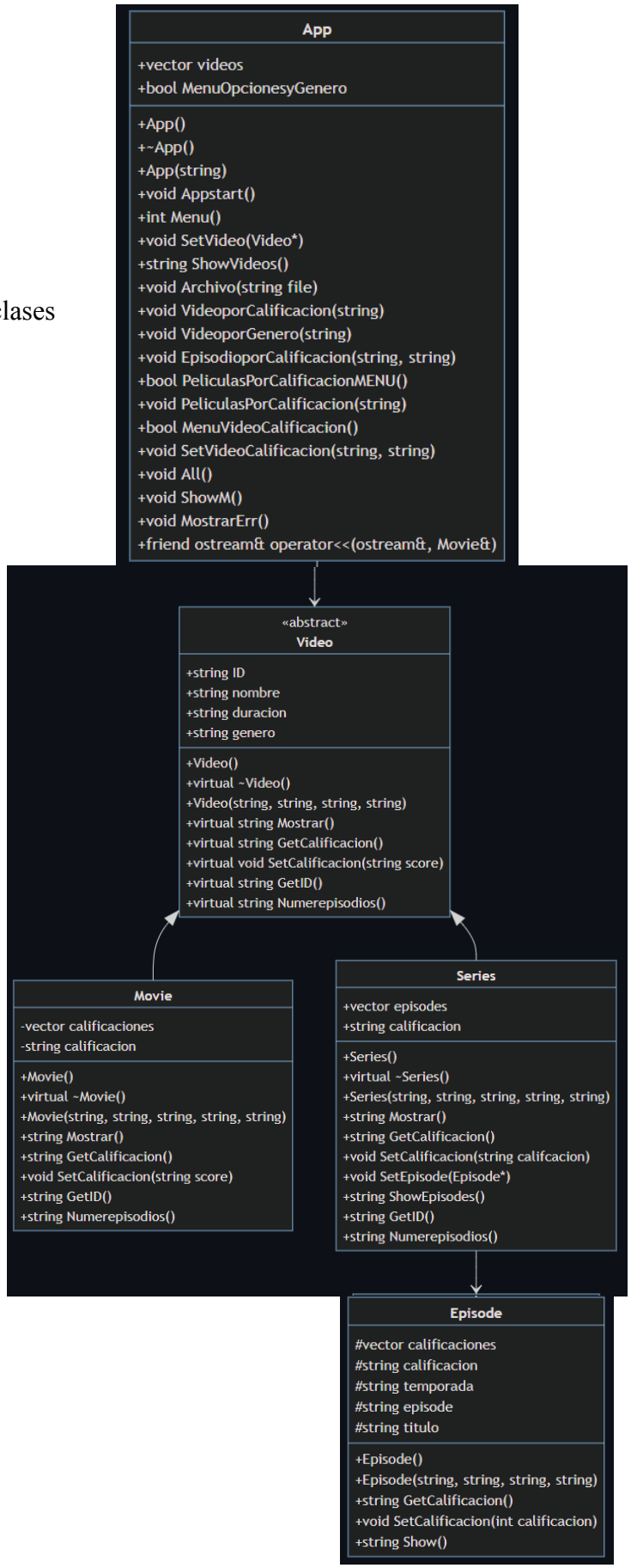
-Mostrar los episodios de una determinada serie con sus calificaciones.

-Mostrar las películas con sus calificaciones (Esta calificación está en escala de 1 a 5 donde 5 es la mejor calificación).

A continuación se mostrarán fragmentos de código para justificar el uso de los conceptos aprendidos en clase para la modelación y/o simulación de un servicio de streaming en el que se muestran películas y series.

El diagrama UML que se mostrará a continuación nos enseña las clases que componen a nuestro servicio de streaming, siendo App la principal, ya que hace uso de todas las clases.

Diagrama de clases UML



La razón por la que decidimos hacer este diagrama fue debido a que creamos un .cpp el cual llamaría a las demás clases. A esta clase la llamamos App, esta mostrará en la pantalla lo que sea requerido, por lo que generamos un constructor con su respectivo destructor. De esta sacamos la siguiente clase, la cual llamamos Video en donde se produce el ID, nombre y calificación de los videos que llama la clase App, por lo que, usamos variables string.

Después, está la clase Movie la cual servirá para la creación de las películas que la clase video nombrará y calificará. Seguidamente, está la clase series la cual funcionará de la misma manera que la clase Movie y, finalmente, la clase Episode la cual será una derivada de la clase Series, la que creará los episodios de la serie que será llamada a la clase App.

Para el diseño de nuestra plataforma de streaming hemos decidido usar un total de 5 clases, de las cuales, como se dijo anteriormente en en el diagrama UML, en donde la clase base que mostrará lo requerido será llamada App y las clases derivadas que crearán los videos, series y películas serán llamadas Video, Movie, Series y Episode.

El concepto de herencia se utiliza en las clases derivadas como lo son Video, Movie, Series y Episodio debido a que comparten sus constructores públicos con la clase base App, ya que esta hace uso de sus funciones públicas para mostrar lo requerido.

```cpp
#pragma once
#include "Video.hpp"
#include "Episode.hpp"


class Series : public Video {
public:
    std::vector<Episode *> episodes;
    std::string calificacion;
public:
    Series();
    virtual ~Series();
    Series(std::string,std::string,std::string,std::string,std::string);
    std::string Mostrar() override;
    std::string GetCalificacion() override;
    void SetCalificacion(std::string calificacion) override;
    void SetEpisode(Episode*);
    std::string ShowEpisodes();
    std::string GetID() override;
    std::string Numerepisodios() override;

};
```

```cpp
#pragma once
#include "Video.hpp"
#include <vector>

class Movie : public Video {
private:
    std::vector<int> calificaciones;
    std::string calificacion;
public:
    Movie();
    virtual ~Movie();
    Movie(std::string,std::string,std::string,std::string,std::string);
    std::string Mostrar() override;
    std::string GetCalificacion() override;
    void SetCalificacion(std::string score) override;
    std::string GetID() override;
    std::string Numerepisodios() override;
};

```

En el caso del concepto de los modificadores de acceso usamos de manera correcta, ya que decidimos poner en públicos los datos y funciones miembros que se van a heredar a las demás clases, como también usamos modificadores protegidos los cuales están limitados a la clase que lo tiene y sus derivadas.

```cpp
#pragma once
#include <string>
#include <vector>

//Todo video tiene un ID, un nombre, una duración y un género (drama, acción, misterio).
class Video {

public:
    std::string ID;
    std::string nombre;
    std::string duracion;
    std::string genero;

public:
    Video();
    virtual ~Video();
    Video(std::string,std::string,std::string,std::string);
    virtual std::string Mostrar()=0;
    virtual std::string GetCalificacion()=0;
    virtual void SetCalificacion(std::string score)=0;
    virtual std::string GetID()=0;
    virtual std::string Numerepisodios()=0;
};
```

Utilizamos de manera correcta el concepto de clases abstractas como se puede ver en el código, esto es debido al uso del método virtual, el cual fue definido junto a la función miembro en la clase base y, posteriormente, es diferida en las clases derivadas con el método override.

```cpp
#pragma once
#include <string>
#include <vector>

//Todo video tiene un ID, un nombre, una duración y un género (drama, acción, misterio).
class Video {

public:
    std::string ID;
    std::string nombre;
    std::string duracion;
    std::string genero;

public:
    Video();
    virtual ~Video();
    Video(std::string,std::string,std::string,std::string);
    virtual std::string Mostrar()=0;
    virtual std::string GetCalificacion()=0;
    virtual void SetCalificacion(std::string score)=0;
    virtual std::string GetID()=0;
    virtual std::string Numerepisodios()=0;
};
```

```cpp
#pragma once
#include "Video.hpp"
#include <vector>

class Movie : public Video {
private:
    std::vector<int> calificaciones;
    std::string calificacion;
public:
    Movie();
    virtual ~Movie();
    Movie(std::string,std::string,std::string,std::string,std::string);
    std::string Mostrar() override;
    std::string GetCalificacion() override;
    void SetCalificacion(std::string score) override;
    std::string GetID() override;
    std::string Numerepisodios() override;
};

```

Se hizo uso de las referencias y apuntadoras, además de polimorfismo. De tal manera que una clase acepta direcciones de objetos de dicha clase y de sus clases derivadas.

```cpp
if (NombreEps == ""){
    Movie *movie = new Movie(IDms,Nombrems,Duracionms,Genrems,califnumms);
    videos.push_back(movie);
} else{
    const bool is_in = conjunto.find(IDms) != conjunto.end();
    if(!is_in){
        Series *serie = new Series(IDms,Nombrems,Duracionms,Genrems,califnumms);
        conjunto.insert(IDms);
        videos.push_back(serie); }
    Series* lastSerie = dynamic_cast<Series*>(videos[videos.size()-1]);
    Episode *episodio = new Episode(Temporadass,califnumms,NombreEps,Califs);
    lastSerie->SetEpisode(episodio);
     }}}
pp::MenuOpcionyGenero(){
```

En este pequeño fragmento del código contenido en App.cpp, se observa un claro ejemplo de polimorfismo dentro if 's. También se observa el uso de apuntadores.

Sobrecargamos únicamente un operador, y es el operador de salida. De tal manera que muestra los datos de una nueva película. Se sobrecargó en una función amiga que se encuentra declarada en la clase App.

```
std::ostream& operator<<(std::ostream& output,Movie *newmovie )
{
    output << newmovie->Mostrar();
    return output;
}
```

Por último, se hizo aplicación del último tema visto en clase que son las excepciones.
Se hizo uso del try/catch para poder mandar un mensaje de error a la terminal, de tal manera que si se digita una opción fuera del rango establecido en el menú, se desplegará un mensaje de error.

```
void App::MostrarErr(){
    try{
        std::cin>>variable;
        if(variable > 7||variable < 0){
            std::string msjegrl { std::to_string(variable) + " no es una variable del menú disponible."};
            throw std::invalid_argument(msjegrl);
        }
    }
    catch(std::invalid_argument &e){
        std::cerr << "Ocurrió algo inesperado: " << e.what() << "\n";

    }
}
```

Y ahora que hemos demostrado el uso de todos estos conceptos en código, se hará una presentación práctica de ello.

```
MENÚ
1: cargar archivo de datos
2: mostrar los videos en general con una cierta calificación o de un cierto género.
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
```

En la imagen se observa el menú que se despliega en la terminal al dar el comando *make run*. Se observan las 6 opciones especificadas en el proyecto, además de 2 nuevas opciones que

permiten ver todos los videos que se encuentran en la aplicación, y otra opción que muestra únicamente las películas.

```
MENÚ
1: cargar archivo de datos
2: mostrar los videos en general con una cierta calificación o de un cierto género.
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
1
Se realizó la subida del archivo con éxito.
```

La opción número uno, simplemente hace una actualización de datos en el archivo input.txt por si se llegaran a agregar nuevas películas o series.

```
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
2
1: mostrar con base a calificaciones.
2: mostrar con base a géneros.
3: regresar.
1
Establezca una calificación: 4
ID movie: 01 Movie: La casa Gucci Duration: 157 Genre: drama Score: 4

ID movie: 05 Movie: Jurassic World Dominion Duration: 147 Genre: acción Score: 4

ID movie: 06 Movie: Jurassic World (2015) Duration: 124 Genre: acción Score: 4

ID movie: 07 Movie: Asesinato en el expreso de oriente Duration: 114 Genre: misterio Score: 4

ID movie: 08 Movie: Siempre a tu lado Duration: 93 Genre: drama Score: 4

ID movie: 09 Movie: Éxodo: Dioses y reyes Duration: 150 Genre: drama Score: 4

ID movie: 10 Movie: Guasón Duration: 122 Genre: drama Score: 4
```

La opción número dos te despliega otras 3 opciones, las cuales son mostrarte series o películas con base en la calificación o rating. En la imagen se muestra un ejemplo de vídeos con base en una calificación de 4.

```
MENÚ
1: cargar archivo de datos
2: mostrar los videos en general con una cierta calificación o de un cierto género.
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
3
Calificación de la serie: 4
ID de la serie: 04
Episode: 2 Title: Chapter Two: The Weirdo on Maple Street Season: 1 Score: 4
```

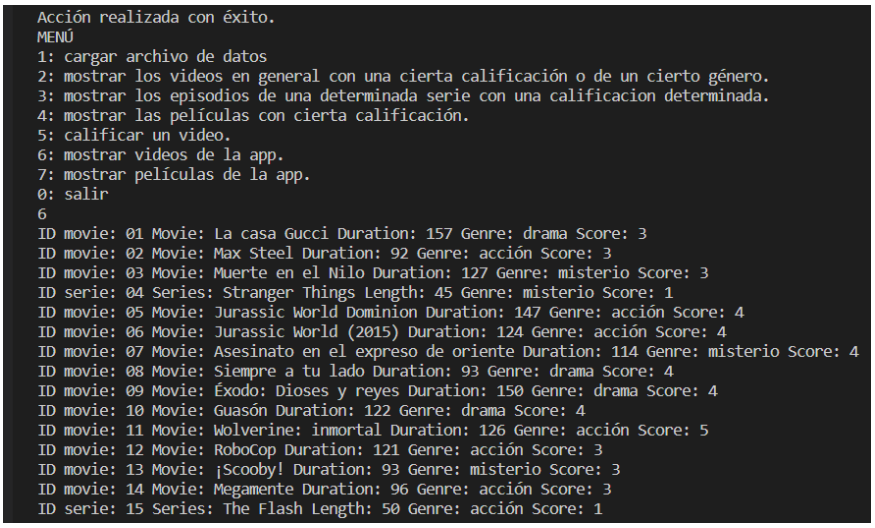
La opción número tres pregunta la calificación de los episodios de alguna serie, para posteriormente preguntarte el ID de la serie que deseas buscar.

```
MENÚ
1: cargar archivo de datos
2: mostrar los videos en general con una cierta calificación o de un cierto género.
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
4
Calificación de las películas: 4
ID movie: 01 Movie: La casa Gucci Duration: 157 Genre: drama Score: 4
ID movie: 05 Movie: Jurassic World Dominion Duration: 147 Genre: acción Score: 4
ID movie: 06 Movie: Jurassic World (2015) Duration: 124 Genre: acción Score: 4
ID movie: 07 Movie: Asesinato en el expreso de oriente Duration: 114 Genre: misterio Score: 4
ID movie: 08 Movie: Siempre a tu lado Duration: 93 Genre: drama Score: 4
ID movie: 09 Movie: Éxodo: Dioses y reyes Duration: 150 Genre: drama Score: 4
ID movie: 10 Movie: Guasón Duration: 122 Genre: drama Score: 4
```

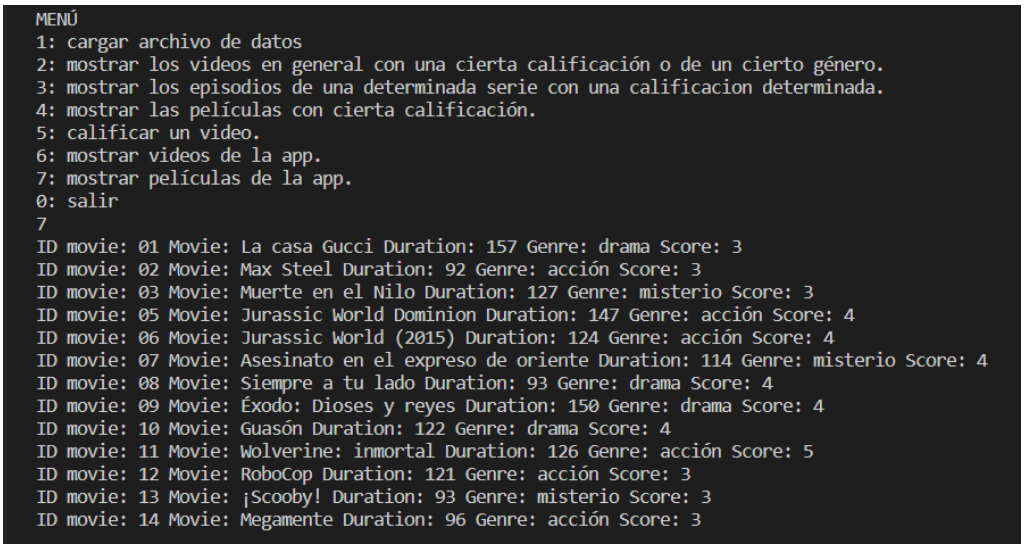
La opción número cuatro muestra las películas de una cierta calificación. En la terminal se le pide al usuario que digite la calificación de las películas que quiere que se le muestre, como un filtro.

```
MENÚ
1: cargar archivo de datos
2: mostrar los videos en general con una cierta calificación o de un cierto género.
3: mostrar los episodios de una determinada serie con una calificacion determinada.
4: mostrar las películas con cierta calificación.
5: calificar un video.
6: mostrar videos de la app.
7: mostrar películas de la app.
0: salir
5
Otorgue el ID del video: 01
Otorgue la calificación: 3
ID movie: 01 Movie: La casa Gucci Duration: 157 Genre: drama Score: 3
```

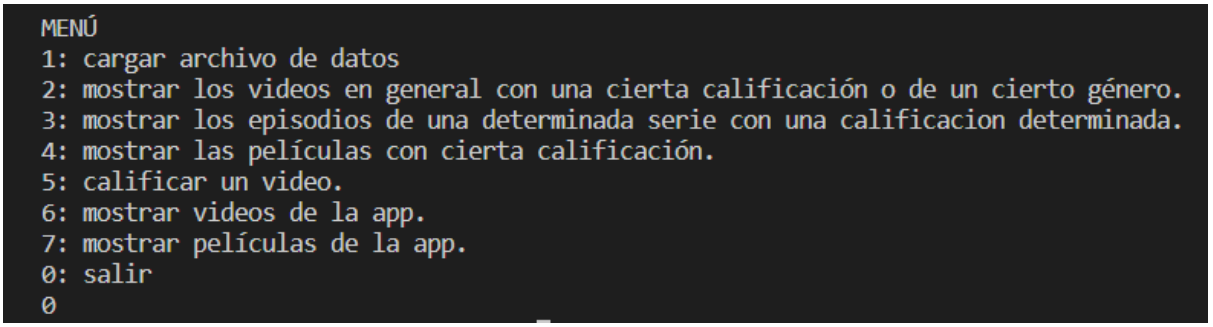
En la opción cinco nos pide el ID de un video y también que le demos una calificación con los parámetros anteriormente definidos(1-5), para establecer una nueva calificación para este video.



En el caso de la opción número seis no muestra todos los videos(movies y series) que existen en la app, dadnos sus respectivas ID, nombres y calificaciones.



Sin embargo en el caso de la opción número siete esta nos muestra lo mismo que en la opción anterior a excepción de que no incluye las series mostrando solamente las películas.



Para finalizar la opción número cero nos permite finalizar el programa terminando su ejecución.

Conclusiones personales

Rogelio:

A lo largo del curso, pude desarrollar mis conocimientos y entender el funcionamiento de la programación orientada a objetos, pues tenía un pensamiento con estructuraciones lineales. Además, pude adquirir nuevas herramientas y realizar las actividades con éxito. Comprendí que programación es una disciplina complicada, pues está muy completa. Finalmente, puedo mencionar que tuve un acompañamiento adecuado de mi profesor, y el proyecto final que realizamos me permitió enriquecer aún más los conocimientos que ya tenía de diversos temas, tales como apuntadores, vectores, clases abstractas, polimorfismo, herencia, downcasting, entre otros.

Julián:

A partir de mi experiencia con el curso, puedo decir que aprendí diversas habilidades del lenguaje c++, ya que antes de empezar con este curso mis conocimientos eran mínimos, sin embargo con el transcurso de la semanas logré aumentar mis habilidades y mi pensamiento crítico con el cual pude discernir cuál es el mejor método a seguir para la resolución de un problema. Por lo tanto pude comprender con mayor medida lo que es el lenguaje de c++ y en este proyecto final puse en práctica todo lo aprendido empleando todos los conceptos vistos en clase como también pude entender un poco más sobre cómo este mundo de la programación orientada a objetos pareciendome muy interesante y con gran aporte a mi formación.

Juan Pablo:

Durante el curso pude darme cuenta de que el lenguaje de c++ no era tan difícil como creía, simplemente requería un entendimiento de los símbolos utilizados en su sintaxis, a partir de ahí, “caminito de la escuela”.

El proyecto final me sirvió para poder hacer un repaso y reafirmación de mi conocimiento sobre programación orientada a objetos. Logrando comprender conceptos como la herencia, las excepciones, apuntadores, polimorfismo, entre otros conceptos. La orientación a objetos en c + + me pareció bastante interesante y divertida, pues nunca me había puesto a pensar acerca de los diferentes componentes que podría tener un objeto, tal como lo fueron los ejemplos vistos en clase sobre un carro, una calculadora y el juego de serpientes y escaleras.

Por último, el proyecto también me hizo darme cuenta de lo difícil que es manejar un servicio de streaming, pues necesitas estar modificando datos en tiempo real, y lo más adecuado sería trabajar con servidores para poder tener un mejor manejo de la información almacenada.

Referencias:

https://github.com/C-SIN-TC1030-301-2211/equipo-2/blob/main/README.md

https://www.geeksforgeeks.org/inheritance-in-c/

https://www.geeksforgeeks.org/pure-virtual-functions-and-abstract-classes/

https://www.geeksforgeeks.org/polymorphism-in-c/

https://www.geeksforgeeks.org/core-dump-segmentation-fault-c-cpp/#:~:text=Core%20Dump%2FSegmentation%20fault%20is,an%20error%20indicating%20memory%20corruption.

https://www.geeksforgeeks.org/vector-insert-function-in-c-stl/#:~:text=std%3A%3Avector%3A%3Ainsert,the%20number%20of%20elements%20inserted.

https://www.geeksforgeeks.org/initialize-a-vector-in-cpp-different-ways/

https://www.geeksforgeeks.org/tokenizing-a-string-cpp/

https://www.geeksforgeeks.org/how-to-sort-an-array-of-dates-in-cc/

https://www.geeksforgeeks.org/pointers-c-examples/