*TDA: LISTA RECOMENDACIONES*

***Lista***:

Archivo principal del programa. Contiene los métodos más genéricos de la lista, que permiten realizar las operaciones elementales y principales del programa. La lista incluye un puntero de tipo nodo, que apunta al primer elemento de la misma y el Tamanio de esta. A medida que se realicen operaciones, estos atributos se irán actualizando, y los puntos apuntarán al nuevo elemento que corresponda. A diferencia de la lista programada anteriormente, en este caso utilizamos plantillas que nos permiten almacenar distintos tipos de datos en la lista sin necesidad de utilizar un puntero genérico de tipo void. Para ello se denomina como un tipo abstracto “Dato” que varía según la necesidad en la función en la cual sea implementado

Atributos:

- Nodo\* primero;

- int tamanio;

Metodos:

//Definición de la plantilla: template <typename Dato>

//Pre: Debe existir un nodo en la posicion pasada por parametro

//Post: Cambia el nodo pasado por parametro por el nodo de la posicion pasada por parametro

- void cambiar\_nodo(Nodo\* nodo,int pos);

//Pre: Debe existir el nodo en esa posicion

//Post: Se va a retirar el nodo de esa posicion

- Nodo<Dato>\* retirar(int pos);

//Descripcion: Constructor

//Pre:---

//Post: Se crea el objeto Lista

+ Lista();

//Pre: El objeto debe existir

//Post: Devuelve el valor de tamanio

+ int obtener\_tamanio();

//Pre: El objeto debe existir, pos debe ser mayor a cero y menor o igual a tamanio

//Post: Se agrego el objeto d a la lista en la posicion pos

+ void agregar(Dato\* d, int pos);

//Pre: El objeto debe existir, pos debe ser mayor a cero y menor o igual a tamanio

//Post: Se eleimino de la lista el objeto en la posicion pos

+ void eliminar(int pos);

//Pre: El objeto debe existir

//Post: Devuelve true si la lista esta vacia y false en caso contrario

+ bool es\_vacia();

//Pre: El objeto debe existir,pos debe ser mayor a cero y menor o igual a tamanio

//Post: Devuelve el elemento de la lista en esa posicion.

+ Dato\* consultar(int pos);

//Pre: El objeto debe existir,pos debe ser mayor a cero y menor o igual a tamanio

//Post: devuelve el nodo enn la posicion pos de la lista

+ Nodo<Dato>\* obtener\_nodo(int pos);

//Pre: El objeto debe existir,pos1 y pos 2 deben ser mayores a cero y menores o iguales a tamanio

//Post: intercambio los nodos en pos1 y pos 2

+ void swap(int pos1, int pos2);

//Pre: El objeto debe existir

//Post: La lista se encuentra ordenada por numero de legajo de los trabajadores

+ void ordenar();

//Pre:El objeto debe existir

//Post:Devuelve verdadero si el dato esta en la lista o falso en caso contrario

bool esta(Dato\* dato\_a\_buscar);

//Descripcion: Destructor

//Pre:El objeto debe existir

//Post:EL objeto es destruido

~Lista();

***Nodo***:

El nodo es quien almacena el objeto que se crea a partir de los datos leídos del archivo de prueba o bien los que carga el usuario por consola, para ello contiene un puntero de tipo “Dato” (que representa a la plantilla) y otro puntero “Siguiente” que apunta al próximo elemento de la lista, dando origen así a lo que llamamos lista simplemente enlazada.

Atributos:

- Dato\* elemento;

- Nodo\* psig;

Metodos:

//Descripcion: Constructor conu un parametro

//Pre:---

//Post: El objeto es creado con los el valor d para elemento y le asigna a psig el valor por defecto

+ Nodo(Dato\* d);

//Descripcion: Constructor con parametros

//Pre:---

//Post: Crea un objeto Nodo

+ Nodo(Dato\* d, Nodo\* psig);

//Pre: El objeto debe existir

//Post: Se modificó el valor de elemento

+ void asignar\_elemento(Dato\* d);

//Pre: El objeto debe existir

//Post: Se modifico el valor de psig

+ void asignar\_siguiente(Nodo\* psig);

//Pre: El objeto debe existir

//Post: Devuelve el valor de elemento

+ Dato\* obtener\_elemento();

//Pre: El objeto debe existir

//Post: Devuelve el valor de siguiente

+ Nodo\* obtener\_siguiente();

//Descripcion: Destructor

//Pre: El objeto debe existir

//Post: El objeto es destruido

+ ~Nodo();

***Película****:*

Clase que representa los objetos que se almacenan en la lista. Inicialmente se crean 2 listas de películas clasificadas como “vistas” y “no vistas” con datos provenientes de archivos de texto. A esas listas el programa las debe analizar y depurar algunas películas que cumplan ciertos requisitos para generar una tercera lista de películas que se consideran “recomendadas”. Así mismo las listas tienen un puntero que apunta a otra lista que en este caso almacena objetos de la clase “Actor” que representan actores de las películas, dando origen al concepto de “listas de listas”

Atributos:

string\* nombre\_pelicula;

- string\* genero;

- int\* puntaje;

- string\* director;

- Lista<Actor>\* actores;

- string\* actores\_s;

Métodos:

//Pre: Existe una lista de actores

//Post: Se agregó un nuevo actor a la lista

- void cargar\_actores(string lectura);

//Descripcion: Constructor sin parametros

//Pre:---

//Post: Se crea el objeto pelicula

+ Pelicula();

//Descripcion: Constructor con parametros

//Pre:---

//Post:Se crea el objeto pelicula

+ Pelicula(string nuevo\_nombre\_pelicula, string nuevo\_genero, string nuevo\_director, int nuevo\_puntaje, string actores\_s\_nuevo);

//Pre: El objeto debe existir

//Post: Devuelve el nombre de la pelicula

+ string obtener\_nombre\_pelicula();

//Pre:El objeto debe existir

//Post:Devuelve el genero de la pelicula

+ string obtener\_genero();

//Pre: El objeto debe existir

//Post: Devuelve el director de la pelicula

+ string obtener\_director();

//Pre: El objeto debe existir

//Post: Devuelve la linea del archivo de peliculas donde estan los actores

+ string obtener\_string();

//Pre: El objeto debe existir

//Post: Devuelve el puntaje de la pelicula

+ int obtener\_puntaje();

//Pre: El objeto debe existir

//Post: Devuelve los actores de la pelicula que fueron cargados

+ Lista<Actor>\* obtener\_actores();

//Pre: El objeto debe existir

//Post: Asigna un nuevo nombre a la pelicula

+ void asignar\_nombre\_pelicula(string\* nuevo\_nombre\_pelicula);

//Pre: El objeto debe existir

//Post: Asigna un nuevo género a la pelicula

+ void asignar\_genero(string\* nuevo\_genero);

//Pre: El objeto debe existir

//Post: Asigna un nuevo director a la pelicula

+ void asignar\_director(string\* nuevo\_director);

//Pre: El objeto debe existir

//Post: Asigna un nuevo puntaje a la pelicula

+ void asignar\_puntaje(int\* nuevo\_puntaje);

//Pre: El objeto debe existir

//Post: Asigna un nuevo actor a la pelicula en la posicion pasada por parametro

+ void asignar\_actor(int pos);

//Pre: El objeto debe existir

//Post: Imprime por pantalla la informacion de la pelicula

+ void imprimir\_informacion();

//Descripcion: Destructor

//Pre: El objeto debe existir

//Post: EL objeto es destruido

~Pelicula();

***Actor:***

Clase que representa a los actores que participan de cada objeto película. Estos se almacenan como atributo de “Película” mediante una segunda lista dentro de la lista de películas

Atributos:

- string\* nombre;

Métodos:

//Descripcion: Constructor

//Pre:---

//Post: Se crea el objeto actor

+ Actor(string nuevo\_nombre);

//Pre: El objeto debe existir

//Post: Devuelve el nombre del actor

+ string obtener\_nombre();

//Pre: El objeto debe existir

//Post: Asigna un nuevo nombre al actor

+ void asignar\_nombre(string\* nuevo\_nombre);

//Descripcion: Destructor

//Pre: El objeto debe existir

//Post: EL objeto es destruido

~Actor();

FUNCIONES\_MAIN:

Archivo header que contiene las definiciones de algunas funciones axiliares programadas de manera estructurada que nos sirven para unir todas las piezas del programa y posteriomente desembocar en el main (Evita que el main tenga infinitas líneas de código)

//Pre: El archivo debe existir y se tiene que poder leer, el objeto pelicula debe existir

//Post: Se va a cargar todos los datos de las peliculas en la lista pasada por parametro

void cargar\_lista(ifstream &archivo,Lista<Pelicula>\* lista);

//Pre: Los objetos pelicula deben existir

//Post: Devuelve true en caso de que 2 peliculas coincidan los nombres de los actores sino devuelve false

bool coinciden\_actores(Pelicula\* pelicula1,Pelicula\* pelicula2);

//Pre: Los objetos pelicula deben existir

//Post: Devuelve true en caso de que 2 peliculas coincidan los nombres de los directores sino devuelve false

bool coinciden\_directores(Pelicula\* pelicula1,Pelicula\* pelicula2);

//Pre: Los objetos pelicula deben existir y las listas ,salvo la de recomendadas, deben estar cargadas

//Post: Crea una nueva lista con las peliculas recomendadas

void crear\_lista\_recomendadas(Lista<Pelicula>\* vistas,Lista<Pelicula>\* no\_vistas,Lista<Pelicula>\* recomendadas);

//Pre:------

//Post: Imprime por pantalla el menu de operaciones

void menu();

//Pre: Los objetos pelicula deben existir y las listas ,deben estar cargadas

//Post: Se va a realizar la oparacion dependiendo del comando pasado por parametro

bool operaciones(char comando,Lista<Pelicula>\* vistas,Lista<Pelicula>\* no\_vistas,Lista<Pelicula>\* recomendadas);