**Universidad Central “Marta Abreu” de Las Villas**

**Facultad de Matemática, Física y Computación**

**Carrera de Ciencia de la Computación**



Proyecto de la Asignatura Programación en C/C++

Título del Proyecto

TASKS MANAGEMENT SYSTEM

**Autor:**

Adriana Marrero Quintero

Diagrama de Clases:

vector<Tarea>

Tarea

- titulo: string

- descripcion: string

- fecha\_vencimiento: string

- prioridad: int

- tarea\_id: unsigned int

- counter: static unsigned int

tarea

Lista\_Tareas

+ filtrar\_Tareas( (\*f) : bool (Tareas) ) : Lista\_Tareas\*

+ mostrarInfo() : void

+ operator < (t1:const Tarea, t2:const Tarea) : bool

Usuario

- nombre: string

- password: size\_t

- lt: Lista\_Tareas

+ addTarea(t: Tarea) : void

+ eliminarTarea(tarea\_id: unsigned int) : bool

+ showInf(i: int) : void

+ filter(prioridad : int) : Lista\_Tareas\*

+ mostrar() : void

Random

Premium

+ addTarea(t: Tarea) : void

+ addTarea(t: Tarea) : void

El Proyecto de Gestión de Tareas es un sistema diseñado para ayudar a los usuarios a organizar y administrar sus tareas de manera eficiente. Con este proyecto, los usuarios pueden crear, editar, eliminar y realizar un seguimiento de sus tareas de una manera estructurada y cómoda.

El proyecto se basa en un conjunto de clases que trabajan juntas para brindar funcionalidades clave.

La clase **Tarea** es una parte fundamental del sistema de gestión de tareas y representa una tarea individual en el programa. Proporciona atributos y métodos para administrar la información de cada tarea, como son:

Atributos:

- titulo: una cadena que almacena el título de la tarea.

- descripcion: una cadena que almacena la descripción de la tarea.

- fecha\_vencimiento: una cadena que almacena la fecha de vencimiento de la tarea.

- prioridad: un entero que representa la prioridad de la tarea.

- tarea\_id: un entero sin signo que actúa como identificador único para la tarea.

- counter: una variable estática que se utiliza para generar identificadores únicos incrementales para cada tarea creada.

Métodos:

- Tarea(): el constructor predeterminado de la clase, que crea una instancia de Tarea sin ningún valor inicial.

- Tarea(const string& tit, const string& descp, const string& fv, int p): un constructor que crea una instancia de Tarea con valores específicos para el título, descripción, fecha de vencimiento y prioridad.

- getTitulo(): devuelve el título de la tarea como una cadena.

- getDescripcion(): devuelve la descripción de la tarea como una cadena.

- getFecha\_Vencimiento(): devuelve la fecha de vencimiento de la tarea como una cadena.

- getPrioridad(): devuelve la prioridad de la tarea como un entero.

- getTarea\_id(): devuelve el identificador único de la tarea como un entero sin signo.

- mostrarInfo(): imprime en la salida estándar la información detallada de la tarea, incluyendo el identificador, título, descripción, fecha de vencimiento y prioridad.

Funciones amigas:

- operator < (const Tarea &t1, const Tarea &t2): sobrecarga del operador < para comparar tareas según su fecha de vencimiento. Esta función amiga permite ordenar las tareas en función de su fecha de vencimiento.

Counter:

- counter es una variable estática de la clase Tarea que se utiliza para generar identificadores únicos para cada tarea creada. Se incrementa cada vez que se crea una nueva tarea.

La clase **Lista\_Tareas**, que se define como una clase derivada de vector<Tarea>, lo que significa que hereda toda la funcionalidad proporcionada por la plantilla de clase vector. En ella se declara:

- una variable privada, tareas, de tipo vector<Tarea> como único atributo.

- un constructor por defecto, Lista\_Tareas(), que no realiza ninguna inicialización específica. Se basa en el constructor por defecto de vector<Tarea> para inicializar el vector subyacente.

- un destructor, ~Lista\_Tareas(), que utiliza la palabra clave default para indicar que se debe utilizar el destructor por defecto proporcionado por el compilador.

- una función filtrar\_Tareas, que toma un puntero a función f como parámetro. El puntero f apunta a una función que toma un objeto Tarea como argumento y devuelve un valor booleano. El propósito de esta función es filtrar las tareas a partir de la función de filtrado proporcionada. La implementación de la función filtrar\_Tareas itera sobre las tareas en el objeto Lista\_Tareas. Para cada tarea, llama a la función de filtrado f y, si devuelve true, agrega la tarea a un nuevo objeto Lista\_Tareas llamado tareas. El objeto Lista\_Tareas filtrado, tareas, se devuelve desde la función.

La clase **Usuario** es una clase abstracta que representa un usuario genérico. Proporciona atributos y métodos para administrar la información del usuario y sus tareas asociadas.

Atributos:

- nombre: una cadena que almacena el nombre del usuario.

- password: un size\_t que almacena la contraseña del usuario. Se utiliza una función hash para convertir la contraseña en un valor numérico seguro.

- lt: un objeto de la clase Lista\_Tareas que almacena las tareas asociadas al usuario.

Métodos:

- Usuario(string n, string p): el constructor de la clase que recibe un nombre y una contraseña para crear un objeto Usuario. Inicializa el atributo nombre con el valor proporcionado, calcula el hash de la contraseña y lo almacena en el atributo password y crea una instancia de Lista\_Tareas y la asigna al atributo lt.

- ~Usuario(): el destructor predeterminado de la clase. No realiza ninguna acción adicional y utiliza el destructor predeterminado para el atributo lt.

- getPassword(): devuelve la contraseña del usuario como un valor de tipo size\_t.

- getNombre(): devuelve el nombre del usuario como una cadena.

- eliminarTarea(unsigned int tarea\_id): elimina una tarea de la lista de tareas del usuario según su identificador único (tarea\_id). Itera sobre la lista de tareas y elimina la tarea que coincide con el identificador proporcionado. Devuelve true si se elimina la tarea con éxito, de lo contrario, devuelve false.

- showInf(int i): muestra la información de las tareas del usuario en la salida estándar. Comienza mostrando la información de la tarea en la posición i de la lista de tareas y luego llama recursivamente a showInf incrementando i en 1, para mostrar la información de la siguiente tarea. Esto se repite hasta que se haya mostrado la información de todas las tareas.

- filter(int prioridad): filtra las tareas del usuario según una prioridad específica. Utiliza una función lambda f para comparar la prioridad proporcionada con la prioridad de cada tarea en la lista de tareas. Devuelve un puntero a una nueva instancia de Lista\_Tareas que contiene solo las tareas que coinciden con la prioridad especificada.

- mostrar(): muestra información básica del usuario, como su nombre y contraseña, en la salida estándar.

Además, el proyecto incluye dos clases especializadas: **Premium** y **Random**.

La clase **Premium** es una subclase de Usuario que hereda sus atributos y métodos, y representa un tipo de usuario específico en el sistema de gestión de tareas.

Métodos:

- Premium(string n, string p): el constructor de la clase que recibe un nombre y una contraseña para crear un objeto Premium. Llama al constructor de la clase padre Usuario con los mismos parámetros para inicializar el nombre y la contraseña del usuario.

- addTarea(Tarea t): agrega una tarea a la lista de tareas del usuario. Verifica si la cantidad de tareas en la lista (lt) es igual a 5. Si es así, no agrega la tarea y simplemente retorna. De lo contrario, agrega la tarea a la lista de tareas.

La clase **Random** también es una subclase de Usuario y hereda sus atributos y métodos.

Métodos:

- Random(string n, string p): el constructor de la clase que recibe un nombre y una contraseña para crear un objeto Random. Llama al constructor de la clase padre Usuario con los mismos parámetros para inicializar el nombre y la contraseña del usuario.

- addTarea(Tarea t): agrega una tarea a la lista de tareas del usuario. Verifica si la cantidad de tareas en la lista (lt) es igual a 3. Si es así, no agrega la tarea y simplemente retorna. De lo contrario, agrega la tarea a la lista de tareas.

El Proyecto de Gestión de Tareas brinda a los usuarios una herramienta eficaz para administrar sus tareas diarias. Con una interfaz intuitiva y funciones personalizadas para diferentes tipos de usuarios, este proyecto busca mejorar la organización y la productividad al facilitar la gestión de tareas de manera efectiva y eficiente.