

|  |
| --- |
| Proyecto Final de C++Fabian Guerra Carrazana 2do Año Ciencia de la Computación. |

# Problemática:

En el almacén de una carpintería se desea implementar un sistema que permita mantener un mejor control de la madera diaria de la que dispone dicha carpintería y el uso que se le da a la misma. La madera se almacena en forma de tablones con igual espesor o grosor(20cm) pero diferente largo y ancho, se conoce que los tablones solo son usados para la confección de muebles, elaboración de artículos decorativos o para encofrado. De todo tablón se conoce su tipo(roble, cedro, pino, etc), el largo y ancho, fecha de entrada al almacén y además cada tablón se identifica por una cadena de texto formado por la letra: M(si es para muebles), D(si es para artículos decorativos) y E(si es para encofrado); un guión medio(-) y también una serie de hasta 4 dígitos(Ejemplo M-1234). De la madera para artículos decorativos se conoce si es fina o no. De la madera para encofrado se conoce además su resistencia(alta, media o baja), de estos tablones se extraen tablas con espesor 4cm y un largo de la mitad del tablón(si el largo del tablón es mayor a los 3m, en caso contrario el largo de la tabla es el mismo que el del tablón).

Por motivos de productividad se desea conocer la ganancia que se espera genere el tablón, la cual se calcula:

* Muebles = Área \* 500

Área es el área de la cara del tablón que no contiene a su espesor

* Decorativos = Área \* (500 +factor\_calidad)

Factor\_calidad = 150(si es fina), 50(si no es fina)

* Encofrado = cantidad\_tablas\_obtenidas \* 200 \* {factor\_resistencia}

factor\_resistencia = 3(si es alta), 2(si es media), 1(si es baja).

El sistema debe se capaz de:

1. Dar de alta a un tablón conociendo su tipo.
2. Calcular la ganancia esperada para un tablón conociendo su identificador. Calcular la ganancia total de los tablones almacenados.
3. Conocer la cantidad de tablones que hay almacenados.
4. Dar salida a un tablón.

# Introducción

En el sector de la carpintería, el control preciso de los recursos, particularmente de la madera, es fundamental para garantizar una producción eficiente y rentable. En este contexto, se plantea la necesidad de implementar un sistema de gestión avanzado en el almacén de una carpintería, con el objetivo de mantener un control detallado de los tablones de madera y optimizar su utilización en la fabricación de muebles, artículos decorativos y encofrado.

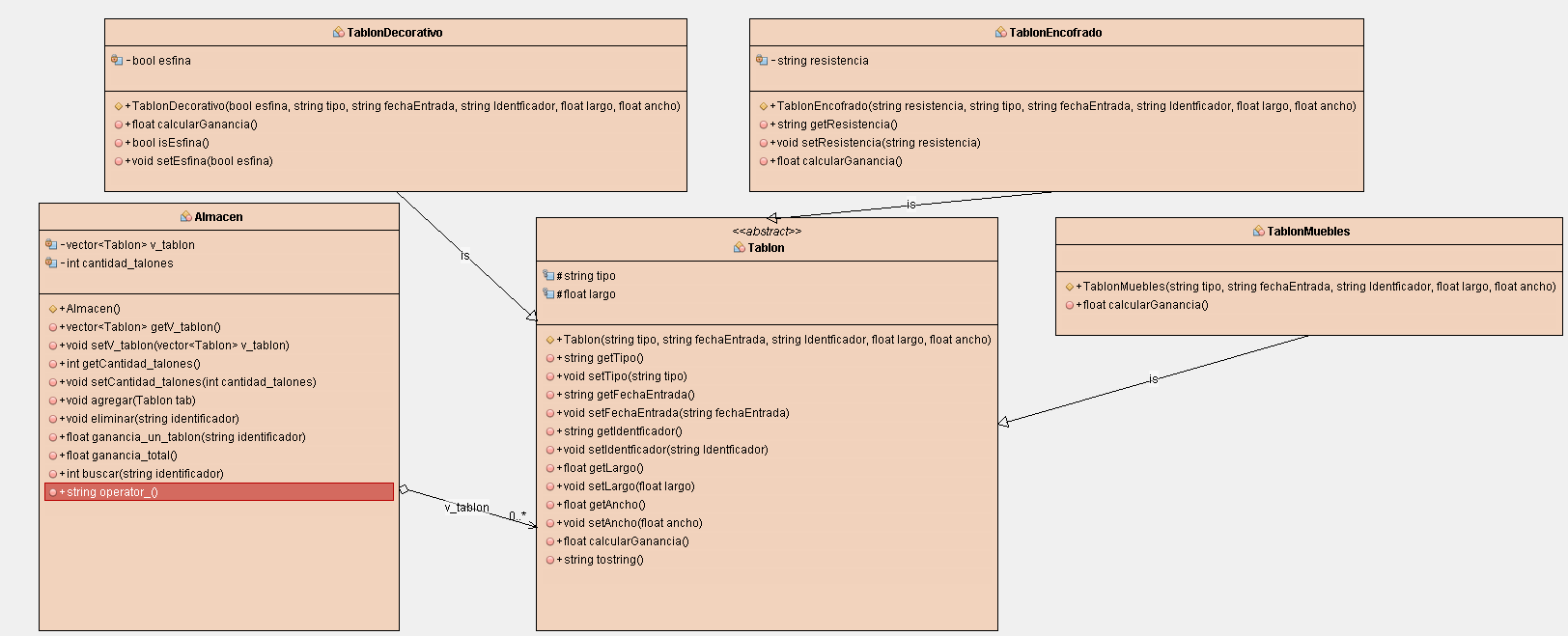
Los tablones de madera, esenciales en los procesos de la carpintería, se almacenan en diferentes formatos y cada uno tiene una función específica según su tipo, dimensiones y destino. Desde robustas piezas para muebles hasta delicados ornamentos decorativos, la diversidad de usos de la madera requiere un sistema eficaz de gestión que permita registrar, clasificar y aprovechar de manera óptima cada tablón.

La identificación y categorización precisa de los tablones, junto con el cálculo proactivo de la ganancia esperada de cada uno, no solo mejora la toma de decisiones en la producción, sino que también contribuye a una planificación estratégica más sólida y a la maximización de los recursos disponibles en el almacén. Por lo tanto, la implementación de un sistema de gestión moderno y estructurado se convierte en un activo invaluable para la carpintería, impulsando la eficiencia operativa y la calidad de los productos finales.

En este contexto, se explorará la aplicación de los principios de la Programación Orientada a Objetos (POO) en C++ para diseñar y desarrollar un sistema de gestión integral de tablones de madera en el almacén de la carpintería. A través de la creación de clases, herencia, polimorfismo y otras técnicas avanzadas de POO, se buscará ofrecer una solución robusta y escalable que satisfaga las necesidades específicas de control, cálculo de ganancias y gestión eficiente de los recursos de madera en el entorno de la carpintería.

# Desarrollo

Una posible solución utilizando POO en C++ para abordar este problema sería diseñar las siguientes clases que se observan en el diagrama:



UML 1

Nota: La herramienta usada para crear el diagrama no me permitió especificar el operador que fue sobrecargado(~) y tampoco permite crear un puntero pues el \* no lo permite colocar. En el código se observan estos aspectos especificados.

1. Clase Tablon: Representaría la entidad de un tablón con atributos como tipo, dimensiones, fecha de entrada y métodos para calcular la ganancia individual(método que se hace virtual puro para implementarse en las clases derivadas correspondientes y además hace abstracta la clase) y también se coloca un método que muestra sus datos.

class Tablon{

protected:

string tipo, fechaEntrada, identificador;

float largo, ancho;

public:

Tablon(string tipo, float largo, float ancho, string fechaEntrada, string identificador);

float getLargo();

float getAncho();

string getTipo();

string getFechaEntrada();

string getIdentificador();

virtual float calcularGanancia()=0;

string tostring();

};

2. Clase TablonMuebles, Clase TablonDecorativo, Clase TablonEncofrado: Clases derivadas de Tablon que específicamente modelan los tablones destinados para muebles, decorativos y encofrado respectivamente, inicializando los tablones con todos sus atributos llamando a su constructor, con métodos para calcular la ganancia según el tipo(se sobrescribe el método CalcularGanancia para cada tipo de tablón).

class TablonMuebles : public Tablon {

public:

TablonMuebles(string tipo, float largo, float ancho, string fechaEntrada, string identificador);

float calcularGanancia() override;

};

class TablonDecorativo : public Tablon{

public:

bool esfina;

TablonDecorativo(string tipo, float largo, float ancho, string fechaEntrada, string identificador, bool esfina);

bool getEsfina();

float calcularGanancia() override;

};

class TablonEncofrado : public Tablon {

private:

string resistencia;

public:

TablonEncofrado(string tipo, float largo, float ancho, string fechaEntrada, string identificador, string resistencia);

string getResistencia();

float cantidad\_tablas();

float calcularGanancia() override;

};

1. Clase Almacen: Contendría una colección de objetos Tablon para gestionar los tablones almacenados, permitiendo dar de alta, calcular ganancia total, conocer la cantidad de tablones almacenados y dar salida a un tablón específico.

class Almacen{

protected:

vector<Tablon\*> v\_tablon;

int cantidad\_tablones;

public:

Almacen();

void setV\_tablon(vector<Tablon\*> v\_tablon);

vector<Tablon\*> getV\_tablon();

void setCantidad\_tablones(int cantidad\_tablones);

int getCantidad\_tablones();

void agregar(Tablon\* tab);

void eliminar(string identificador);

float ganancia\_un\_tablon(string identificador);

float ganancia\_total();

int buscar(string identificador);

string operator ~();

};

Conclusiones

La implementación de un sistema de gestión de tablones de madera en una carpintería utilizando Programación Orientada a Objetos (POO) en C++ representa un paso significativo hacia la mejora de la eficiencia operativa y la optimización de los recursos en el proceso de producción carpintera.

Al aplicar los principios de POO, como la creación de clases, la herencia y el polimorfismo, se logra una modelización clara y modular de los tablones y sus características, lo que facilita el desarrollo, mantenimiento y expansión del sistema. La jerarquía de clases permite representar de manera precisa y estructurada la diversidad de tablones de madera, cada uno destinado a un propósito específico en la carpintería.

La utilización de métodos virtuales y la capacidad de sobrescribir funciones en las clases derivadas permiten adaptar el comportamiento de los tablones de acuerdo con sus usos específicos, como la confección de muebles, la elaboración de artículos decorativos o el encofrado. Este enfoque posibilita un cálculo preciso de la ganancia esperada de cada tablón, contribuyendo a una toma de decisiones basada en datos y a una planificación más efectiva de la producción.

Además, el diseño modular y flexible del sistema de gestión de tablones en la carpintería, basado en POO, establece una base sólida para futuras mejoras y adaptaciones en función de las necesidades cambiantes del negocio. La estructura ordenada y la escalabilidad inherente del enfoque POO en C++ brindan una solución duradera y adaptable que puede evolucionar con la carpintería a lo largo del tiempo.

En resumen, la incorporación de la Programación Orientada a Objetos en el desarrollo del sistema de gestión de tablones de madera en una carpintería abre nuevas oportunidades para optimizar los procesos, aumentar la productividad y mejorar la calidad de los productos finales, consolidando así la eficacia y la competitividad del negocio en el mercado.