INFORME

Desafío #2 Informática 2

Estudiante: Cristian David Martinez De La Ossa.

CC. 1010024054

Profesor: Aníbal Guerra

Institución: Universidad de Antioquia.

Fecha: 12/10/2024

Informe del Proyecto: Sistema de Comercialización de Combustible TerMax

Introducción

Este informe detalla el desarrollo de un sistema de gestión para una red nacional de estaciones de servicio de combustible. La solución fue planteada utilizando los principios de la programación orientada a objetos (POO) en C++. A lo largo del desarrollo, he buscado diseñar un modelo que represente adecuadamente las dinámicas de una empresa de distribución de combustibles, optimizando tanto la estructura del programa como la eficiencia de las operaciones.

1. Contexto y análisis del problema

El objetivo principal del proyecto es desarrollar un sistema que permita la gestión eficiente de una red nacional de estaciones de servicio de combustible. Las estaciones de servicio están distribuidas en tres regiones del país (norte, centro y sur) y cada una de ellas cuenta con varios surtidores para la venta de tres tipos de combustible: Regular, EcoExtra y Premium. Además de gestionar las ventas, el sistema debe permitir agregar y eliminar estaciones, controlar los precios del combustible, simular ventas y verificar la existencia de fugas de combustible.

Mi análisis empezó por identificar los principales elementos a modelar: las estaciones de servicio, los surtidores y los tanques de combustible. Cada uno de estos elementos interactúa entre sí, lo que implica la necesidad de encapsular las relaciones y las operaciones en clases adecuadamente diseñadas.

2. Diseño del Sistema

Para modelar este problema en C++, utilicé los principios de la programación orientada a objetos (POO), dividiendo las responsabilidades entre varias clases:

- **Clase Gasolinera:** Esta clase representa cada estación de servicio, la cual contiene información sobre su nombre, código, gerente, región, coordenadas y tanques de

combustible. Los surtidores son parte integral de las gasolineras y están gestionados por esta clase.

- **Clase surtidor:** Cada gasolinera cuenta con entre 2 y 12 surtidores, y esta clase se encarga de modelar los surtidores individuales. Cada surtidor registra las ventas por tipo de combustible y actualiza el nivel del tanque al realizar una venta.
- **Clase TerMax:** Es la clase principal que controla la red de gasolineras. En esta clase, he implementado funciones para gestionar las estaciones a nivel nacional, tales como agregar nuevas estaciones, eliminar estaciones sin surtidores activos, y calcular el monto total de ventas por categoría de combustible.

La relación entre las clases está bien definida. Cada gasolinera tiene varios surtidores, y TerMax mantiene un arreglo de gasolineras para cada región (norte, centro y sur). Para cada una de estas regiones, se implementó un sistema de redimensionamiento dinámico que permite ajustar la capacidad de almacenamiento de gasolineras conforme se agregan o eliminan estaciones.

3. Implementación

A lo largo del desarrollo, implementé varias funciones clave para gestionar las estaciones de servicio y los surtidores:

- **Agregar y eliminar gasolineras:** Utilicé arreglos dinámicos para almacenar las gasolineras de cada región, y las funciones agregarGasolinera y eliminarEstacion gestionan la adición y eliminación de estaciones. En caso de que se alcance la capacidad máxima de un arreglo, el sistema redimensiona el arreglo dinámicamente para permitir la inclusión de más estaciones.
- **Simulación de ventas:** Cada surtidor registra ventas por categoría de combustible (Regular, EcoExtra, Premium). La función simularVenta permite al usuario realizar una transacción, donde se calcula la cantidad de combustible vendida, se actualiza el tanque de la estación y se imprime un resumen de la venta. La función también

gestiona el estado del surtidor (activo o inactivo) dependiendo de la disponibilidad de combustible.

- **Cambio de precios:** En función de la región, los precios del combustible pueden ser ajustados para todas las estaciones de una región específica. Esto asegura que los precios sean consistentes en cada región, y permite al usuario modificar los precios según sea necesario.
- **Gestión de memoria dinámica:** Implementé un manejo cuidadoso de memoria al utilizar arreglos dinámicos. Cada vez que se modifica el tamaño del arreglo de gasolineras en una región, la memoria se gestiona correctamente para evitar fugas de memoria. Esto fue crucial en la implementación, dado que las estaciones de servicio y los surtidores están en constante expansión o eliminación.

4. Retos y Soluciones

Durante el desarrollo, uno de los retos más importantes fue manejar correctamente la memoria dinámica. Al utilizar arreglos dinámicos para almacenar las gasolineras y los surtidores, tuve que asegurarme de redimensionar los arreglos de manera eficiente, liberando la memoria cuando ya no se necesitaba. Inicialmente, enfrenté problemas con la duplicación de código en la gestión de las gasolineras para cada región. Sin embargo, refactoricé algunas funciones para simplificar el código y reducir la redundancia.

Otro reto fue la implementación de la simulación de ventas, ya que requería ajustar correctamente los niveles de los tanques y gestionar las transacciones dependiendo de la cantidad de combustible disponible. Para resolver esto, incluí validaciones que permiten que la venta solo se realice si hay suficiente combustible, y si no es posible, se limita la venta a lo que esté disponible en el tanque.

Finalmente, fue necesario considerar la variabilidad en la cantidad de surtidores por estación y la necesidad de ajustar los precios de manera consistente en cada región. Esto se resolvió mediante el uso de bucles que aseguran que todas las estaciones de una región tengan precios uniformes.

5. Evolución del Proyecto

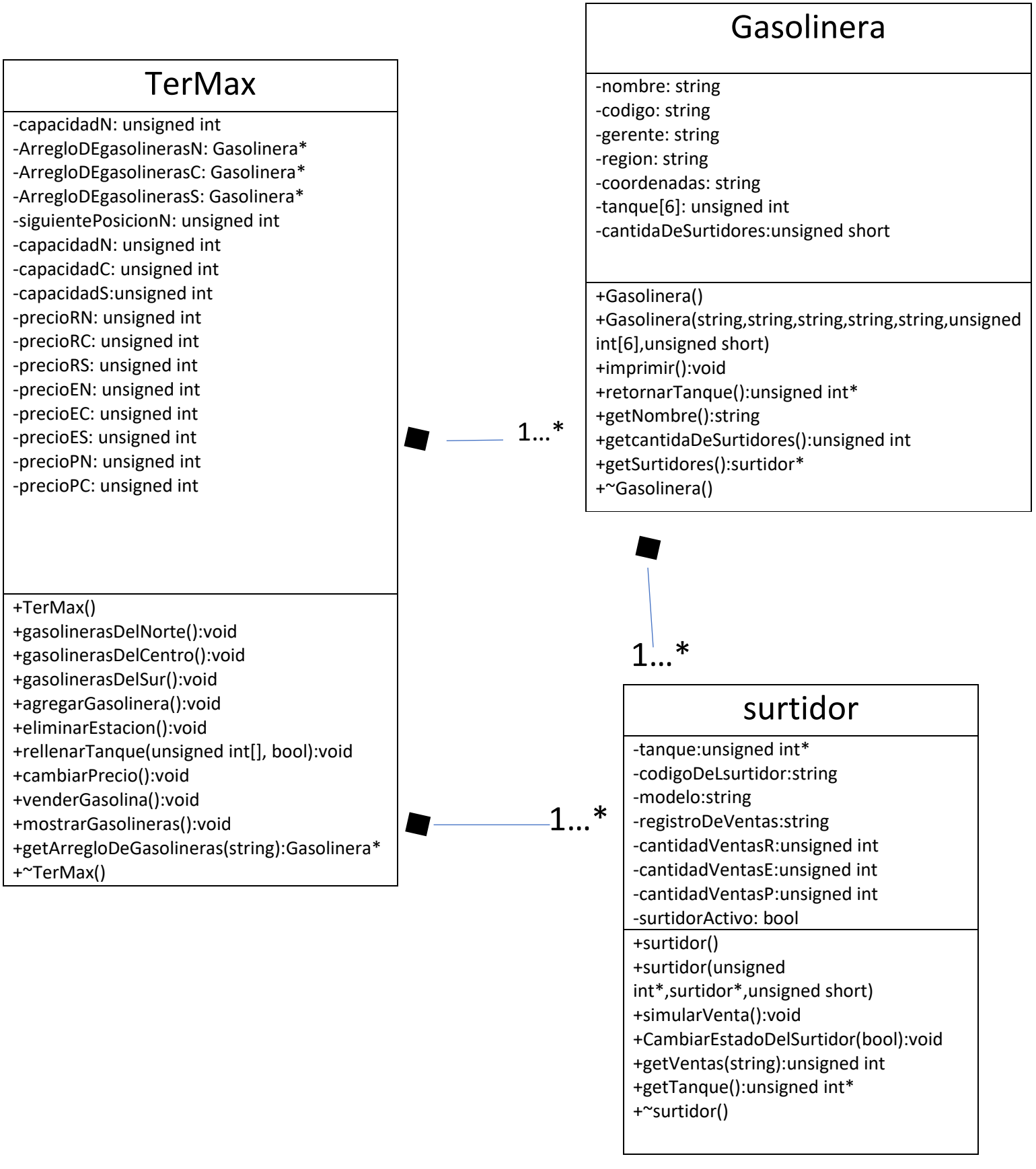
Hasta ahora, he implementado las funcionalidades esenciales del sistema, incluyendo la gestión de estaciones de servicio, la simulación de ventas y el manejo de precios. Sin embargo, aún quedan por implementar algunas funcionalidades importantes, como la verificación de fugas de combustible. Esta funcionalidad implicará la comparación entre lo vendido y lo almacenado en el tanque, asegurando que la diferencia no exceda el 5% de la capacidad original del tanque.

En las próximas etapas, también debo agregar la funcionalidad para gestionar los surtidores individualmente, permitiendo activar o desactivar surtidores, así como consultar su histórico de transacciones.

Conclusión

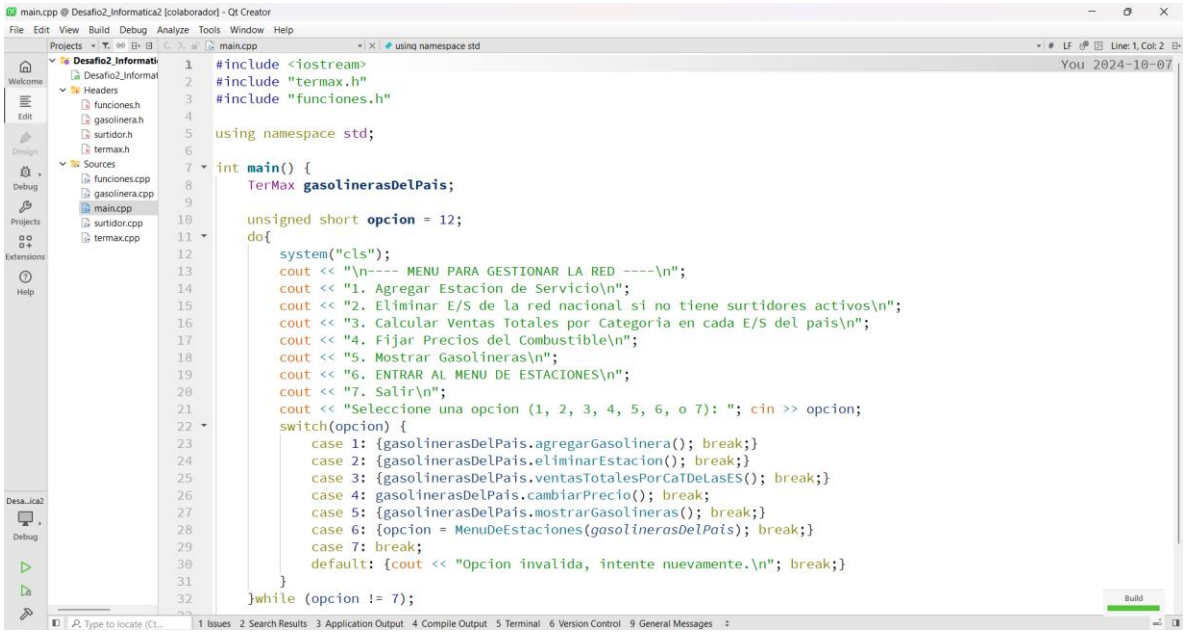
El desarrollo del sistema de comercialización de combustible TerMax ha sido un proceso iterativo, en el cual he tenido que ajustar el diseño y la implementación para asegurar la eficiencia y la correcta gestión de memoria. A través de este proyecto, he podido aplicar los principios de la programación orientada a objetos en C++ para resolver un problema complejo de la vida real, y he logrado desarrollar una base sólida que me permitirá completar el proyecto con éxito.

Diagrama de Clases UML Simplificado



- Gasolinera tiene una relación de composición con surtidor, lo que implica que los surtidores no pueden existir sin una gasolinera asociada.
- TerMax tiene una relación de composición con Gasolinera, lo que implica que las Gasolineras no pueden existir sin TerMax.
- TerMax tiene una relación de composición con surtidor, lo que implica que los surtidores no pueden existir sin TerMax, pues sin TerMax no habría Gasolineras y por tanto tampoco surtidores.

ANEXOS



surtidor.h @ Desafio2_Informatica2 [colaborador] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects | surtidor.h | using namespace std | LF | Line: 1, Col: 1

Desafio2_Informati
- Desafio2_Informat
- Headers
  - funciones.h
  - gasolinera.h
  - surtidor.h
  - termax.h
- Sources
  - funciones.cpp
  - gasolinera.cpp
  - main.cpp
  - surtidor.cpp
  - termax.cpp

```cpp
class surtidor
{
private:
    unsigned int* tanque;
    string codigoDeLsurtidor, modelo, registroDeVentas = "";
    unsigned int cantidadVentasR = 0, cantidadVentasE = 0, cantidadVentasP = 0, saldoVentasR = 0, saldoVentasE = 0, sald
    bool EstadoDelSurtidor;
    surtidor* DireccionDeSurtidores;
    unsigned short cantidaDeSurtidores;
public:
    surtidor();

    surtidor(unsigned int* DireccionTanque, surtidor* _DireccionDeSurtidores, unsigned short _cantidaDeSurtidores);

    surtidor(const surtidor &Acopiar);

    void simularVenta(unsigned short modo, string cambiarDia);
    void CambiarEstadoDelSurtidor(bool estado);
    bool VerEstadoDelSurtidor();
    unsigned int getSaldoVentas(string categoria);
    unsigned int getCantidadVentas(string categoria);
    unsigned int* getTanque();
    string getcodigoDeLsurtidor();
    string getRegistroDeVentas();


    void imprimirAtributosDelSurtidor();

    ~surtidor();
};
```

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Terminal 6 Version Control 9 General Messages

gasolinera.h @ Desafio2_Informatica2 [colaborador] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects | gasolinera.h | using namespace std | LF | Line: 1, Col: 1

Desafio2_Informati
- Desafio2_Informat
- Headers
  - funciones.h
  - gasolinera.h
  - surtidor.h
  - termax.h
- Sources
  - funciones.cpp
  - gasolinera.cpp
  - main.cpp
  - surtidor.cpp
  - termax.cpp

```cpp
#ifndef GASOLINERA_H                                                    You 2024-10-08
#define GASOLINERA_H
#include <string>
#include "surtidor.h"
using namespace std;

class Gasolinera
{
private:
    surtidor surtidores [12];
    unsigned short cantidaDeSurtidores;
    unsigned long int totalSaldoVentas = 0;
    unsigned long int loQueQuedaEnElTanque = 0;
    unsigned int capacidadDelTanque; float TotalR, TotalE, TotalP;

    string nombre, codigo, gerente, region, coordenadas;
    unsigned int* tanque = new unsigned int [6];
    bool liberarTanque = false;
public:
    Gasolinera();
    Gasolinera(string nombre, string _codigo, string _gerente, string _region, string _coordenadas, unsigned int _tanque
    void imprimir();
    unsigned int* getTanque();
    string getNombre();
    unsigned short getcantidaDeSurtidores(string modificar = "NO");
    void liberarElTanque();
    unsigned long int actualizarTotalVentas();
    void historialTransaciones();
    unsigned int imprimirLitrosVendidos(bool imprimir = true);
    unsigned int verCapacidadDelTanque(unsigned int nuevoValor = 0);
    void detectarFugas();
```

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Terminal 6 Version Control 9 General Messages