**INFORME: Desafío 2. Informática 2**

**Diana Lucia Baeza Ruiz, Sebastián Giraldo Álvarez**

1. **Análisis del problema**

Se necesita diseñar un sistema de gestión para una red de estación de servicio de combustible utilizando POO, que permita un manejo eficiente de las estaciones de servicio, los surtidores y las transacciones de venta de combustible.

Para esto necesitamos asegurarnos que el sistema administre de manera eficiente los siguientes aspectos:

* Estaciones de servicio a nivel nacional
* Surtidores
* Disponibilidad de combustible en el tanque central
* Transacciones de venta del combustible en sus tres categorías:
* Regular
* Premium
* EcoExtra
* Registro de ventas de cada surtidor

Para dicha implementación se requiere identificar y diseñar las clases, los atributos y las operaciones que debe cumplir el sistema:

1. Estación de servicio: cada una se identifica con un nombre, código identificador, gerente, región, ubicación geográfica (en GPS) y tanque central donde se almacena las tres categorías (Regular, Premium, EcoExtra) cuya capacidad varia por categoría y estación.
2. Tanque central: almacena las diferentes categorías, se encarga de actualizar la cantidad de combustible disponible cada vez que un surtidor vende.
3. Surtidores: cada uno debe tener un código identificador, un modelo y distribuir las tres categorías. Debe registrar las ventas (con fecha, hora, cantidad categoría y método de pago) y ajustar el inventario al tanque central.
4. Ventas: Incluye los detalles de cada transacción, sobre la cantidad de combustible, categoría, método de pago (efectivo, Tdebito, Tcredito) y cantidad cobrada.
5. Cliente: almacena información básica del cliente (número de documento).

El sistema debe cumplir con la gestión de estaciones de servicio como añadir, modificar, eliminar estaciones y gestionar la ubicación y capacidad de almacenamientos y el numero de surtidores de cada una.

Actualizar automáticamente la cantidad de combustible disponible en cada tanque después de realizar una venta y asegurarse de que no se cobre mas de lo entregado en caso de no estar disponible la cantidad de combustible solicitada.

Registrar las ventas con detalles de fecha, hora, cantidad y tipo de combustible, incluyendo el método de pago, indicando desde que estación y surtidor se hace esta.

1. **Consideraciones para la solución propuesta**

Las clases deben estar relacionadas entre sí para que puedan interactuar con los diferentes componentes:

* Cada E/S tiene un tanque central y puede tener entre 2 y 12 surtidores.
* Cada surtidor se conecta a un tanque central y registra las ventas de combustible en sus tres categorías.
* Las ventas afectan directamente la disponibilidad de combustible al tanque central.
* Las estaciones se agrupan por región.

Definición de cada clase y sus atributos:

1. Clase Red Nacional de Estaciones De Servicio **(RedEstaciones):**

Atributos:

* Numero de estaciones
* Cantidad máxima de estaciones
* Precio de combustible Regular
* Precio de combustible Rremium
* Precio de combustible EcoExtra
* Litros vendidos de combustible Regular
* Litros vendidos de combustible Premium
* Litros vendidos de combustible EcoExtra
* Estaciones de Servicio (arreglo dinámico)

Métodos:

* Abrir base de datos: Para obtener la información de las estaciones guardad en una base de datos de un archivo .txt.
* Crear estación de servicio: Para almacenar la información de cada estación en un arreglo dinámico y poder iterar en ellas.
* Eliminar estación
* Calcular el monto total de las ventas por cada categoría
* Guarda la base de datos, para luego de procesar la información de estaciones y surtidores la información se actualice
* Convertir una cadena de texto a número entero
* Convertir una cadena de texto a número decimal
* Separar una cadena de texto

1. Clase Estación De Servicio:

Atributos:

* Número de surtidores
* Cantidad máxima de surtidores
* Surtidores (arreglo dinámico con los surtidores de cada estación)
* Estado del surtidor (Activo/Inactivo)
* Nombre
* Gerente
* Región
* Longitud
* Latitud
* Cantidad de combustible Regular
* Cantidad de combustible Premium
* Cantidad de combustible EcoExtra
* Identificador

Métodos:

* Crear un surtidor
* Eliminar un surtidor
* Activar/Desactivar Surtidor
* Verificar las Capacidades de cada categoría de combustible

1. Clase Surtidor:

Atributos:

* Estado del surtidor
* Numero de ventas máximas por surtidor
* Código identificador
* Registro Ventas (Arreglo dinámico)
* Modelo del surtidor

Métodos:

* Vender
* Desactivar surtidor
* Activar surtidor

1. **Diagrama de clases de la solución planteada**
2. **Algoritmos implementados**

En la clase RedEstaciones implementamos algoritmos que están diseñados para gestionar una red de estaciones de servicio. Tomando los datos inicialmente de un archivo de texto (BaseDeDatos.txt), permitiendo entonces crear, eliminar y analizar los datos de las estaciones y sus surtidores. Para lo cual se describen a continuación los métodos implementados:

1. SepararString: Se encarga de dividir una cadena de texto en fragmentos o bloques mas pequeños basados en dos delimitadores. Busca las posiciones de los delimitadores y extrae los bloques de texto que hay en medio de estos almacenándolos en un arreglo. Se usa para procesar las líneas que son leídas de la base de datos de estación y surtidores.
2. convertirAEntero: Toma una cadena de texto que representa un número entero y lo convierte a un valor de tipo short int. Recorre cada carácter de la cadena, lo multiplica por 10 y sumando el valor numérico del carácter (Tabla ASCII).
3. convertirAFloat: Es un proceso parecido al anterior, pero aquí se convierte a un valor del tipo float, teniendo en cuenta el punto decimal y adicionalmente verifica si el numero es negativo.
4. abrirBaseDatos: Con este algoritmo se realiza la tarea mas importante, que es abrir la BaseDeDatos, procesando la información línea por línea, separando los datos en bloques y a partir de aquí comienza a crear las estaciones y los surtidores correspondientes. También registra las ventas realizadas en los surtidores organizando toda la información en las estructuras indicadas.
5. crearEstacion: Permite la creación de nuevas estaciones de servicio, solicitando al usuario los datos necesarios. Verifica que no se haya alcanzado el limite de estaciones, valida las capacidades de combustible y asigna un identificador secuencial. Si las entradas son válidas, se crea la estación y se agrega al final del arreglo estaciones.
6. eliminarEstacion: Elimina una estación de servicio solicitando solo el identificador de la estación. Antes de esto, verifica que la estación no tenga surtidores activos, de no ser así no se permite la eliminación. Si no hay surtidores activos, se elimina la estación y se reorganiza el arreglo.
7. calcularMontoTotalPorCategoria: El algoritmo recorre las estaciones y sus surtidores para calcular el total de ventas por categoría de combustible, mediante el registroVentas de los surtidores, lo procesa y realiza los cálculos multiplicando la cantidad de litros vendidos por su correspondiente precio. Al final indica el monto total y la cantidad total de litros.
8. guardarBaseDatos: Guarda la información de las estaciones, los surtidores y el registro de ventas en el mismo formato en el que se leyó la información de la primera base de datos. Guardando en la primera línea los precios del combustible, de la segunda en adelante la información de cada estación seguida de la información de surtidores y su registro de ventas, cada línea corresponde a una estación.

En la clase EstacionDeServicio hace referencia a una estación de servicio de la red nacional de estaciones, guardando información propia como su identificación, nombre, región, gerente, ubicación y la capacidad máxima que puede almacenar por cada categoría de combustible, así como sus surtidores asociados. Los algoritmos clases implementados son:

1. Constructor: Inicializa los atributos de la estación, incluyendo un arreglo de punteros a objetos Surtidor para representar los surtidores. Utilizando un for para inicializar todos los punteros a nullptr, indicando que inicialmente no hay ninguno.
2. crearSurtidor: crea un nuevo objeto Surtidor y los agrega al arreglo de surtidores, teniendo en cuanta que cuando se alcanza el máximo de surtidores, se reemplazan los más antiguo.
3. eliminarSurtidor: Con el código identificador del surtidor a eliminar, lo busca en el arreglo dinamico, libera la memoria del objeto y desplaza los elementos siguientes hacia atrás.
4. activarDesactivarSurtidor: con el código identificador busca en el arreglo de puntero y le cambia el estado. Si el estado es true/activo, lo invierte a false/inactivo.
5. verificarCapacidades: compara la cantidad solicitada de combustible con la disponibilidad en la estación, retorna true si hay suficiente de lo contrario es false.

En la clase Surtidor, que representa un surtidor de gasolina individual dentro de una estación, además de guarda información básica como identificador y modelo, permite realizar ventas y hacer un registro de cada una de ellas, para lo que se implementaron los siguientes algoritmos:

1. Constructor: inicializa los atributos del surtidor, con el código identificador, modelo y estado, además inicializa un puntero a string con el registro de ventas en 0 y todos los punteros a nullptr.
2. Vender: Construye una cadena de texto con la información de la venta, concatenando los datos proporcionados, asigna la cadena a la siguiente posición disponible en el arreglo y en caso de que se alcance el máximo de ventas entonces se sobrescriben en las ventas más antiguas.
3. **Problemas afrontados durante el desarrollo de la solución**

* Puesto que decidimos implementar el uso de arreglos dinámicos para almacenar la información de cada estación, los surtidores y las ventas, se necesitaba asegurarse de que cuando se necesitara eliminar una de estas, se hiciera correctamente la liberación en memoria.
* Se nos dificulto el manejo de excepciones en todos los casos requeridos, tener en cuenta que cuando se pide ingresar datos estos correspondan a la información solicitada, como en el caso de registrar ventas, que no verifica si la fecha o la hora ingresada son correctas, puesto que se almacenan como cadena de esto, lo cual no hace ninguna verificación correspondiente.
* No logramos implementar la verificación de fugas de combustible.
* Aunque los surtidores registran las ventas, no se verifica si la cantidad solicitada esta disponible o no en el tanque central, la venta se hace o se registra, aunque la cantidad solicitada supere a la disponible.
* El número de estaciones y el número de surtidores esta definido de manera limitada, para el numero de estaciones, ya tenemos 30 en la base de datos solo se pueden crear 5 más, lo suficiente para saber que se crean de la manera adecuada, para los surtidores, el máximo es de 12, entonces inicializa cada estación con dos surtidores activo.
* Para el registro de ventas se debe tener en cuenta que el máximo que se pueden crear son 5 por cada surtidor.

1. **Consideraciones para la implementación**

* El número de estaciones y el número de surtidores está definido de manera limitada, para el número de estaciones, ya tenemos 30 en la base de datos solo se pueden crear 5 más, lo suficiente para saber que se crean de la manera adecuada, para los surtidores, el máximo es de 12, entonces inicializa cada estación con dos surtidores activo.
* Para el registro de ventas se debe tener en cuenta que el máximo que se pueden crear son 5 por cada surtidor.
* La opción del menú que permite registrar ventas no tiene ninguna excepción con los datos que se le ingresan, tener en cuenta la cantidad de litros solicitada si sea un numero para que al momento de hacer la conversión a entero no tenga problemas