**Documentación del código**

**1 - Gestión de red:**

### **Descripción General**

La clase **gestionRed** gestiona operaciones relacionadas con estaciones de servicio de combustible, incluyendo la adición, eliminación, fijación de precios y cálculo de ventas. La clase utiliza métodos de la clase **funcionesApoyo** para realizar diversas operaciones, como leer y escribir archivos, seleccionar datos, y calcular montos de ventas.

### **Métodos Públicos**

1. **gestionRed()**
   * Constructor de la clase. Inicializa un objeto de tipo **gestionRed**.
2. **void agregarEstacion(funcionesApoyo &fa)**
   * Agrega una nueva estación de servicio.
   * Genera automáticamente un código, selecciona un gerente, asigna una región y genera coordenadas GPS.
   * También genera cantidades aleatorias de combustible para Regular, Premium y EcoExtra.
   * La estación recién agregada se muestra y se guarda en el sistema.
3. **void eliminarEstacion(funcionesApoyo &fa)**
   * Elimina una estación especificada por su ID.
   * Verifica que la estación existe y que no tiene islas activas.
   * Elimina la estación, así como sus islas y transacciones asociadas, y actualiza los arreglos correspondientes.
   * Decrementa el contador total de estaciones.
4. **void calcularVentasPorEstacion(funcionesApoyo &fa)**
   * Calcula las ventas de combustible para cada estación.
   * Para cada estación, determina los precios según la región (Norte, Sur, Centro).
   * Suma los litros vendidos por categoría (Regular, Premium, EcoExtra) y calcula el monto total de ventas.
5. **void fijarPreciosCombustible(funcionesApoyo &fa)**
   * Establece nuevos precios para cada región.
   * Solicita al usuario los precios para las tres categorías de combustible (Regular, Premium, EcoExtra) en las regiones Norte, Sur y Centro.
   * Actualiza los precios en el archivo y en el arreglo correspondiente.
6. **void fijarPreciosCombustible(double precioNorte[3], double precioSur[3], double precioCentro[3])**
   * Sobrecarga de la función **fijarPreciosCombustible**.
   * Permite fijar los precios de combustible utilizando arreglos de tres elementos que representan los precios para Regular, Premium y EcoExtra en cada región.

### **Métodos Privados**

1. **double sumarMontosPorCategoria(const string &idEstacion, const string &categoria)**
   * Calcula la suma de los montos por una categoría específica de combustible en una estación.

### **Uso de la Clase funcionesApoyo**

La clase **gestionRed** depende de la clase **funcionesApoyo** para:

* Leer y escribir archivos (gerentes, precios, estaciones).
* Generar datos automáticos (códigos, coordenadas GPS).
* Realizar cálculos (monto de ventas).
* Manejar la estructura de datos (estaciones, islas, transacciones).

### **Notas**

* La memoria dinámica utilizada para los arreglos de códigos y nombres de gerentes es liberada después de su uso.
* La eliminación de una estación incluye la eliminación de las islas y transacciones asociadas.
* Se asegura que no se puedan agregar estaciones si no hay gerentes disponibles o si no se selecciona un gerente válido.

### **Consideraciones de Error**

* En la función **agregarEstacion**, si no hay gerentes disponibles, se muestra un mensaje de error y se detiene la operación.
* En **eliminarEstacion**, si el ID proporcionado no corresponde a una estación existente o la estación tiene islas activas, se muestra un mensaje de error y no se realiza la eliminación.

Esta documentación resume la funcionalidad del código y la interacción con otros componentes del sistema, como los archivos de datos y las clases auxiliares.

**2 - Simulación de ventas y sistema nacional de verificación de fugas**

**Descripción General**

La clase **simulacionVentas** se encarga de simular las ventas de combustible en estaciones de servicio y de verificar posibles fugas en los tanques de las estaciones. Utiliza la clase **funcionesApoyo** para manejar tareas auxiliares, como seleccionar estaciones, determinar precios, realizar cálculos de ventas, y gestionar las transacciones.

### Métodos Públicos

1. **simulacionVentas**()  
   Constructor de la clase. Inicializa un objeto de tipo **simulacionVentas** sin parámetros específicos.
2. **int simularVenta(funcionesApoyo &fa, gestionRed &gr)**  
   Realiza una simulación de venta de combustible en una estación.
   1. Selecciona una estación de servicio, el tipo de gasolina, y una isla al azar.
   2. Genera una cantidad aleatoria de litros vendidos y determina el precio por litro según la región de la estación y el tipo de gasolina.
   3. Calcula el monto total de la venta y obtiene la fecha y hora actuales.
   4. Pide al usuario el método de pago y el ID del usuario, validando este último.
   5. Registra la transacción en el sistema y muestra un mensaje de confirmación.
3. **void verificarFugas(funcionesApoyo &fa)**  
   Verifica si hay fugas en los tanques de una estación de servicio seleccionada.
   1. Obtiene las capacidades de los tanques de la estación para Regular, Premium, y EcoExtra.
   2. Suma las ventas de cada tipo de combustible para la estación.
   3. Compara el nivel actual de cada tanque con el total de ventas y la capacidad del tanque para determinar si existen fugas.
   4. Muestra el estado de los tanques y las posibles fugas detectadas.

### Uso de la Clase **funcionesApoyo**

La clase **simulacionVentas** depende de la clase **funcionesApoyo** para las siguientes tareas:

* Seleccionar estaciones, islas y tipos de gasolina.
* Obtener y validar datos del usuario.
* Realizar cálculos de montos de ventas.
* Registrar transacciones en el sistema.
* Verificar los niveles de los tanques y detectar fugas.

### Consideraciones de Error

* En el método **simularVenta**, si la selección de la estación o la isla no es válida, la operación se detiene y se muestra un mensaje de error.
* Si la región de la estación o el tipo de gasolina no son válidos, se notifica al usuario y la simulación no se completa.
* En el método **verificarFugas**, si no se encuentra la estación seleccionada, se muestra un mensaje de error y no se realiza la verificación.

**3 - Funciones de Apoyo**

### **Descripción General**

La clase **funcionesApoyo** proporciona varias funcionalidades para la gestión de datos relacionados con estaciones de combustible, islas de servicio, transacciones y precios de combustible en una aplicación de gestión de estaciones de servicio. La clase incluye métodos para leer datos de archivos, mostrar información, validar usuarios, seleccionar opciones de combustible y región, generar códigos y coordenadas, procesar ventas y manejar archivos de respaldo.

### **Métodos Públicos**

* **funcionesApoyo()**
  + Constructor que inicializa los contadores de estaciones, islas, transacciones, y establece el estado de gerente en falso.
* **void leerArchivo(const string &archivo)**
  + Lee un archivo de texto que contiene información sobre estaciones, islas y transacciones, y los organiza en las matrices correspondientes.
* **void leerArchivoPrecios(const string &archivoPrecios)**
  + Lee un archivo que contiene los precios de los combustibles para distintas regiones y los almacena en la matriz **precios**.
* **void mostrarDatos()**
  + Muestra la información de todas las estaciones, islas y transacciones registradas.
* **void mostrarPrecios()**
  + Muestra los precios de los combustibles (Regular, Premium, EcoExtra) para cada región con dos decimales.
* **bool loginUsuario()**
  + Método que verifica si un usuario tiene acceso al sistema.
* **bool esGerente() const**
  + Devuelve **true** si el usuario actual es un gerente, de lo contrario devuelve **false**.
* **void mostrarMenuAdministrador() / mostrarMenuEmpleado()**
  + Muestran el menú correspondiente para el administrador o empleado.
* **void mostrarSubMenuGestionRed() / mostrarSubMenuGestionEstaciones() / mostrarSubMenuVerificacionFugas() / mostrarSubMenuSimulacionVentas()**
  + Muestran submenús para gestionar la red de estaciones, estaciones individuales, verificar fugas y simular ventas, respectivamente.
* **string seleccionarTipoGasolina()**
  + Permite al usuario seleccionar el tipo de gasolina (Regular, Premium, EcoExtra) y devuelve la selección.
* **string generarCodigoEstacion()**
  + Genera un código de estación secuencial basado en la cantidad actual de estaciones.
* **string seleccionarRegion()**
  + Permite al usuario seleccionar una región (Norte, Centro, Sur, Otra).
* **string generarCoordenadasGPS()**
  + Genera coordenadas GPS aleatorias dentro de un rango específico.
* **int generarCantidadCombustible()**
  + Genera una cantidad aleatoria de combustible entre 100 y 200.
* **void agregarEstacion() / agregarTransaccion()**
  + Agrega una nueva estación o transacción a los arreglos correspondientes.
* **void armarNuevaTransaccion()**
  + Crea una nueva transacción con la información proporcionada.
* **void guardarDatos(const string &nombreArchivo) / guardarPrecios(const string &archivoPrecios)**
  + Guarda los datos de estaciones y precios en archivos respectivos.
* **void transferirNuevaEstacion() / transferirNuevaTransaccion()**
  + Transfiere los datos de nuevas estaciones o transacciones a las matrices principales.
* **int contarGerentes(const string& nombreArchivo)**
  + Cuenta el número de gerentes registrados en un archivo.
* **bool identificarEstacion(string &idEstacion)**
  + Verifica si un ID de estación es válido.
* **double calcularMontoVenta(double litrosVendidos, double precioCombustible)**
  + Calcula el monto de una venta multiplicando los litros vendidos por el precio del combustible.
* **string trim(const string &str)**
  + Elimina los espacios en blanco al inicio y final de una cadena.
* **string addPadding(const string &str)**
  + Añade un espacio adicional al inicio y final de una cadena.

### **Métodos Privados**

* **void procesarEstaciones(const string &linea)**
  + Procesa una línea de texto que contiene información de una estación y la almacena en la matriz **estaciones**.
* **void procesarIslas(const string &linea)**
  + Procesa una línea de texto con información de una isla.
* **void procesarTransacciones(const string &linea)**
  + Procesa una línea de texto con detalles de una transacción.
* **void procesarPrecios(const string &linea)**
  + Procesa una línea de texto con los precios de los combustibles para una región.

### **Atributos**

* **Matrices de almacenamiento**
  + **estaciones, islas, transacciones, precios:** matrices fijas que almacenan información leída y procesada.
  + **newEstacion, newIsla, newTransaccion:** matrices para agregar datos nuevos.
* **Contadores**
  + **totalEstaciones, totalIslas, totalTransacciones:** controlan la cantidad de registros almacenados.

### **Constantes**

* **MAX\_ESTACIONES, MAX\_ISLAS, MAX\_TRANSACCIONES**: límites superiores para la cantidad de datos almacenados.
* **MAX\_COLUMNAS\_E, MAX\_COLUMNAS\_I, MAX\_COLUMNAS\_T**: límites de columnas en los arreglos.