Documentación del Proyecto

Título del Proyecto

Sistema de Gestión de Parqueadero de Vehículos

Descripción del Proyecto

El proyecto consiste en el desarrollo de un sistema de gestión de parqueadero, en el cual los usuarios pueden gestionar la entrada y salida de vehículos en un parqueadero de capacidad limitada. El sistema está diseñado para permitir registrar, retirar y consultar vehículos, así como mantener un historial de los vehículos que han estado en el parqueadero. Además, utiliza archivos de texto para almacenar y persistir los datos entre las distintas ejecuciones del programa.

Este sistema incluye opciones de validación de datos de entrada (como placa y cédula), visualización de los puestos del parqueadero y gestión del historial de vehículos.

Autores

- **Autor 1**: Gualotuña Amaguaya Brayan Patricio
- **Autor 2**: Guayasamin Heredia Josue Daniel
- **Autor 3**: Lisintuña Correa Cristian Mateo

Fecha

Fecha de Creación: 25/11/2024

Requisitos Funcionales

1. **Ver parqueadero**:
    - El sistema debe permitir al usuario ver el estado actual del parqueadero. Esto incluye una representación gráfica de los puestos ocupados y libres, con información de los vehículos estacionados en cada puesto. El número de puestos disponibles debe ser mostrado en tiempo real.
2. **Ingresar vehículo**:

    - Permite al usuario registrar un vehículo que llega al parqueadero. El sistema solicita la placa del vehículo, el número de cédula del propietario, y sus nombres y apellidos completos.
    - El sistema valida que la placa siga un formato específico (3 letras seguidas de 4 dígitos) y que la cédula sea válida (10 dígitos numéricos).
    - Si el parqueadero tiene un puesto libre, el vehículo se asigna a un puesto. Si el parqueadero está lleno, el sistema muestra un mensaje informando que no hay espacio.
3. **Retirar vehículo**:
    - El usuario puede retirar un vehículo del parqueadero seleccionando el número de puesto en el que está estacionado el vehículo. El sistema libera el puesto y actualiza la disponibilidad del parqueadero.
    - Los datos del vehículo retirado se registran en un historial de vehículos, para un posterior seguimiento.
4. **Ver Datos**:
    - Permite visualizar los datos de los vehículos estacionados actualmente en el parqueadero. Esta información incluye la placa, el nombre del propietario, cédula y el puesto donde está estacionado el vehículo.
5. **Ver Vehículos**:
    - El sistema debe permitir ver una lista completa de los vehículos que han ingresado al parqueadero, ya sea actualmente o en el pasado. Se debe mostrar la placa, cédula, nombre del propietario y otros detalles relevantes.
6. **Ver Historial**:
    - Muestra un historial detallado de los vehículos que han ingresado y salido del parqueadero en el pasado. Este historial debe incluir la fecha y hora de ingreso y retiro, junto con los detalles del vehículo.
7. **Salir**:
    - La opción para salir del sistema. Al seleccionar esta opción, el sistema debe finalizar correctamente y guardar todos los datos persistentes antes de salir.

Requisitos No Funcionales

1. **Interfaz de usuario**:
    - La interfaz de usuario es completamente en consola, con menús interactivos. El usuario debe poder navegar entre las opciones utilizando las teclas de flecha hacia arriba y hacia abajo, y seleccionar una opción con la tecla Enter.
    - El diseño debe ser claro y fácil de usar, con colores para resaltar las opciones seleccionadas.
2. **Persistencia de datos**:
    - Todos los datos relacionados con los vehículos, como placas, cédulas, historial y estado del parqueadero, deben ser almacenados en archivos de texto. Esto asegura que los datos persistan entre sesiones del programa.
    - Los archivos de texto deben ser gestionados de manera eficiente y deben permitir la lectura y escritura rápida.
3. **Validación de datos**:

    - El sistema debe verificar que las entradas del usuario, como la placa, cédula y otros datos relevantes, sean válidas antes de aceptar la información. Esto incluye asegurarse de que la placa siga el formato correcto y que la cédula tenga 10 dígitos numéricos.
    - Si el usuario ingresa datos incorrectos, el sistema debe mostrar un mensaje de error y pedir al usuario que intente nuevamente.
4. **Rendimiento**:
    - El sistema debe ser capaz de manejar operaciones con un número limitado de vehículos, ya que el parqueadero tiene un número máximo de 20 puestos.
    - Las operaciones de ingreso, retiro y visualización de datos deben realizarse rápidamente y sin errores.
5. **Compatibilidad del sistema**:
    - El sistema está diseñado para ejecutarse en sistemas operativos Windows, utilizando bibliotecas específicas como `<windows.h>` para manipular la consola y `<conio.h>` para manejar entradas de teclado en tiempo real.

Entorno de Desarrollo

- **Lenguaje de programación**: C++
- **Entorno de desarrollo**: [IDE utilizado] (Ejemplo: Visual Studio, Code::Blocks, etc.)
- **Sistema operativo**: Windows (El código hace uso de bibliotecas específicas de Windows como `windows.h` y `_getch()`)
- **Dependencias y bibliotecas**:
    - **`conio.h`**: Para capturar entradas del usuario sin mostrar texto en la consola (funciones como `_getch()` para lectura de caracteres).
    - **`windows.h`**: Para cambiar el color del texto en la consola mediante la función `SetConsoleTextAttribute()`.
    - **`iostream`**: Para la entrada y salida estándar de datos (funciones `cin`, `cout`).
    - **`string`**: Para el manejo de cadenas de texto.
    - **`cctype`**: Para validaciones de caracteres (como `isalpha()` y `isdigit()`).

Descripción del Código

Funciones Principales

1. **Función `menu()`**:
    - Esta función muestra un menú interactivo en la consola donde el usuario puede elegir qué acción realizar. Las opciones del menú son:
        - Ver parqueadero
        - Ingresar vehículo
        - Retirar vehículo

        - Ver Datos
        - Ver Vehículos
        - Ver Historial
        - Salir
    - El menú se navega utilizando las teclas de flecha y se selecciona una opción con Enter. El programa ejecuta la acción correspondiente a la opción seleccionada.
2. **Función `mostrarMenu()`**:
    - Esta función es responsable de dibujar el menú en la pantalla, resaltando la opción seleccionada por el usuario y permitiendo la navegación mediante las teclas de flecha.
3. **Función `ingresar_cedula()`**:
    - Permite al usuario ingresar su número de cédula, validando que se ingresen exactamente 10 dígitos numéricos. Si el usuario presiona una tecla incorrecta, se emite un sonido de error (beep).
4. **Función `validarPlaca()`**:
    - Valida que la placa del vehículo esté en el formato correcto: 3 letras mayúsculas seguidas de 4 números. Si la placa no sigue este formato, se solicita nuevamente al usuario que ingrese una placa válida.
5. **Función `ingresar_string()`**:
    - Esta función se usa para ingresar cadenas de texto, como el nombre y apellido del propietario del vehículo. Solo se permiten letras y espacios, y se asegura de que al menos una letra sea ingresada.
6. **Función `procesarSeleccion()`**:
    - Esta función ejecuta la acción correspondiente a la opción seleccionada en el menú. Dependiendo de la opción, el sistema puede:
        - Mostrar el estado actual del parqueadero.
        - Registrar un nuevo vehículo.
        - Retirar un vehículo.
        - Ver los datos de los vehículos estacionados.
        - Ver el historial de vehículos.
7. **Clase `ListaCircularDoble`**:
    - Esta clase es la responsable de gestionar el parqueadero. Utiliza una estructura de lista doblemente circular para almacenar los vehículos. Cada nodo en la lista representa un puesto de estacionamiento y contiene los datos del vehículo, si está ocupado.
    - Permite realizar operaciones como ingresar vehículos, retirar vehículos y mostrar los puestos libres y ocupados.
8. **Clase `Archivotxt` y `Archivotxt2`**:
    - Estas clases se encargan de la persistencia de datos en archivos de texto. Usan métodos como `leerPlacas()`, `guardarDatos()`, `guardarPlacas()` y `guardarHistorial()` para leer y escribir la información de los vehículos en archivos, asegurando que los datos sean almacenados entre las ejecuciones del programa.

Pruebas Unitarias

Las pruebas unitarias se centran en verificar que cada función del sistema funcione correctamente y que las entradas y salidas sean las esperadas.

1. **Prueba de validación de placa**:
    - Entrada: "ABC1234"
        - **Salida esperada**: `true` (Placa válida)
    - Entrada: "A123456"
        - **Salida esperada**: `false` (Placa inválida)
2. **Prueba de validación de cédula**:
    - Entrada: "1234567890"
        - **Salida esperada**: `true` (Cédula válida)
    - Entrada: "123456789"
        - **Salida esperada**