Registro Y Control De Vehículos

Universidad Nacional Experimental de Guayana
Técnicas De Programación II
Jesús David Pérez
Yanexi Sánchez
E-mail: jesusdavid2405 @gmail.com
yanexi.sanchez95@gmail.com
Febrero 8 del 2015
Ciudad Guayana, Edo. Bolívar

Resumen

El Código fuente ha sido desarrollado en CodeBlocks bajo "Lenguaje C", se trabajó con una estructura de árbol binario ordenado y un archivo binario para cargar y guardar toda la estructura del árbol.

Palabras Claves: Código Fuente, CodeBlocks, Archivos.

1. Introducción

Un organismo oficial encargado del tránsito terrestre, desea llevar el control de los vehículos que circulan por el país, para ello debe mantener la información de los vehículos de manera permanente en un archivo en disco. Para realizar y agilizar todas las operaciones básicas sobre el archivo (agregar, buscar, eliminar, etc.) de manera optima, estas acciones se realizaran de manera dinámica, a través de un árbol binario ordenado por la placa del vehículo (la cual es única).

2. Planteamiento del problema

- Elaborar un programa para el registro y control de vehículos que almacene en disco: placa del vehículo, cedula del propietario, nombre del propietario, marca, modelo, año, color, serial de motor, dirección e-mail del propietario. El programa debe guardar y cargar desde un archivo binario una estructura de árbol binario.

3. Resolución del problema:

Se pretende elaborar el registro y control de vehículos de la siguiente manera:

- Tener un archivo donde cargar y guardar la estructura de árbol binario llamado "Registro De Vehículos".
- Tener un menú con todas las opciones posibles para el funcionamiento del programa.
- Elaborar una función que verifique y convierta todas las letras minúsculas de las placas ingresadas en letras mayúsculas, para

guardarse uniformemente en el registro.

- Elaborar una función que crea un nodo de tipo árbol y devuelva dichos campos de la estructura inicializada.
- Elaborar una función para pedir todos los datos tanto de los propietarios como de los vehículos.
- Elaborar una función que permita verificar que la placa cumpla con el formato del control de vehículos.
- Elaborar una función que valide el signo "@" ingresado cuando se obtenga el correo del propietario.
- Elaborar una función que inserte el nodo creado a nuestra estructura de árbol binario manteniendo la relación de menores a la izquierda y mayores a la derecha a través de la placa del vehículo.
- Mostrar un sub-menú de búsqueda con todas las posibles opciones, para luego proceder con una búsqueda a profundidad con el dato ingresado.
- Elaborar una función que busque específicamente el parámetro introducido por el usuario en toda la estructura de árbol binario.
- **4. Contenido**

El registro, inicia con una pequeña presentación de introducción que contiene una grafica, la cual tiene una duración de 20 segundos aproximadamente conjunto con un mensaje. Posteriormente, se presenta el menú principal donde se observan las diferentes opciones del programa.

5. Marco teórico

Es importante señalar las librerías utilizadas en el código fuente del registro, y a su vez describir la utilización de los recursos y programas que dieron a facilitar la exitosa ejecución del mismo.

Con respecto a las librerías, se utilizaron las siguientes: <stdio.h>,

<conio.h>, <ctype.h>, <stdlib.h>, <string.h>, <windows.h>

Entre las funciones más utilizadas con las librerías ya nombradas, tenemos:

- ✓ Fflush(stdin); para vaciar el búfer del teclado.
- ✓ If(); para realizar una o varias instrucciones según sea la condición.
- ✓ Switch-case-break-default; para ejecutar un conjunto de instrucciones según sea el caso de la respuesta obtenida.
- ✓ For ( ){}, while ( ); son estructuras repetitivas para llevar a cabo uno o varios tipos de instrucciones, se repite según la condición definida.
- ✓ Strcpy; hace una copia de una cadena de caracteres y se le asigna a otra variable.
- ✓ System("color "); utilizada para cambiar el color de la pantalla y el color de las letras.
- ✓ System("cls"); para borrar o limpiar la pantalla.
- ✓ Printf (""); para imprimir cualquier mensaje por pantalla.
- ✓ Sleep ( ); utilizada para esperar cierto tiempo en la ejecución del programa, definiendo el tiempo a esperar.
- ✓ Gotoxy ( );: posiciona el cursor en un lugar especifico en la pantalla. Esta función necesita de dos argumentos que le

indican las coordenadas de la pantalla hacia donde mover el cursor.

- ✓ Strcmp ( ); La función strcmp compara las cadenas hasta el carácter situado "\0".
- ✓ Strcpy ( ); Copia la cadena apuntada por origen en la cadena apuntada por destino.

Funciones principales del registro:

- ✓ **Void** cargar_barra ( ); Carga una barra con caracteres de la tabla ascii.
- ✓ **Void** menú ( ); Esta función muestra un menú en pantalla para que el usuario pueda acceder al registro.
- ✓ **ARBOL*** crear_nodo ( ); Función que crea un nodo.
- ✓ **void** agregar (ARBOL* &p); Función que permite a los usuarios agregar más nodos al registro.
- ✓ **char** placas (char *cadena); Función que convierte las letras minúsculas de una placa en letras mayúsculas.
- ✓ **bool** verificar (ARBOL* &p, char *cadena); Función que verifica que la placa ingresada existe o no en el árbol binario.
- ✓ **bool** insertar (ARBOL* &p, char *placa); Función que permite insertar un nodo al árbol.
- ✓ **void** pedir_datos (ARBOL* &p, char *placa); Función que pide los datos para iniciar el registro.
- ✓ **bool** verificar_correo (char *correo); Función que verifica que el correo contenga un signo "@".
- ✓ **void** mostrar_registro (ARBOL *p); Función que permite mostrar el registro de las placas en el árbol en postorden.
- ✓ **void** borrar_registro (ARBOL* &p); Función que permite borrar todo el registro del árbol.
- ✓ **void** eliminar (ARBOL* &p); Función que verifica que registro se va a eliminar.
- ✓ **bool** eliminar_registro (ARBOL* &p); Función que permite eliminar el registro (NODO) del árbol.
- ✓ **void** guardar_registro (ARBOL* &p); Función que permite guardar el registro en el archivo.
- ✓ **void** menu_buscar(ARBOL* &p); Función que muestra un menú para elegir por cual parámetro desea buscar.
- ✓ **bool** buscar (ARBOL* &p, char *dato, int resp); Función que permite buscar el parámetro ingresado en el árbol.
- ✓ **void** mostar_datos (ARBOL *p); Función que permite mostrar los datos del registro.
- ✓ **void** modificar (ARBOL* &p); Función que verifica si la placa existe para modificar el registro en el nodo.
- ✓ **ARBOL*** modificar_datos (ARBOL* &p, char *placa); Función que busca la placa en el árbol y modifica el registro en el nodo.
- ✓ **char** guardar_registro (FILE *registro, ARBOL* &p); Función que permite guardar el árbol en el archivo.
- ✓ **Void** cargar_archivo (FILE *registro, ARBOL* &p, char *placa); Función que permite cargar un archivo al árbol.

6. Pruebas

Se realizaron diversas pruebas a lo largo de la elaboración del programa, estas pruebas corresponden a las presentaciones de los menús y su funcionamiento correcto, obteniendo así respuestas satisfactorias para

la continuidad del registro. No obstante, donde hubo mayor énfasis fue en las repetitivas pruebas del funcionamiento de la búsqueda por parámetros personalizada y en la creación y carga del archivo binario.

En referencia a este último, es importante señalar que no se logró hacer que la estructura de árbol binario cargara desde el archivo binario. Sin embargo, falto rediseñar la función eliminar ya que solo elimina exitosamente sí el nodo es una hoja.

Nota: ”Cuando se termina de ejecutar el programa, este automáticamente genera el archivo binario llamado Registro De Vehiculos.dat el cual guarda toda la estructura del árbol binario. Es importante señalar que cuando este se vuelva a ejecutar se intenta acceder al registro para cargar el árbol, esta función no logra cargar dicho archivo, lo cual es el motivo de que el ejecutable.exe deje de funcionar”.

7. Conclusión

La implementación de una estructura de árbol binario en un programa es una parte fundamental en la aplicación de múltiples tareas en el mismo, ya que facilita la búsqueda y mantiene un orden por definición, donde se ubican los datos menores a la izquierda y los mayores a la derecha. Este programa fue de vital importancia esta estructura de dato ya que en toda y en su absoluta aplicación se conseguía resolver los casos a través del manejo de dicha estructura con funciones recursivas, lo que significativamente se obtiene como resultado una buena simplificación de código, ya que manejarlo iterativamente se nos reflejaría algo largo y en casos, muy pesado la lectura del mismo. No obstante, para finalizar hay que señalar que el manejo de archivos cumple también un papel importante en este rol, ya que a través de él escribimos, leemos e incluso guardamos cualquiera información que necesitáramos, en este programa de registro y control de vehículos utilizamos el archivo como una herramienta de soporte o base para garantizar todo el registro de la estructura de árbol binario durante la ejecución del programa.