UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructura de Datos B

Ing. Carlos Gustavo Alonzo

Aux. Josué David Itzep Salvador

PROYECTO #2

CSSiga

OBJETIVOS

General:

- Transformar los conocimientos teóricos aprendidos acerca de las estructuras de datos, en soluciones prácticas, y efectivas, que resuelvan problemas de la vida cotidiana

Específicos

- Desarrollar habilidades en el área de aplicaciones web
- Comprender el manejo de memoria en distintos ambientes de desarrollo
- Comprender y administrar el funcionamiento de estructuras de datos tanto lineales como no lineales, así como algoritmos de encriptado.

Descripción

La compañía encargada de la gestión de transporte público CSSiga, ha solicitado a los estudiantes de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala que desarrollen un sistema para mejorar el control de las transacciones que giran alrededor del sistema de transporte público de la ciudad de Guatemala. Esta compañía está a cargo del control, asignación de rutas a los distintos autobuses y pilotos de los mismos, así como el control de transacciones de los usuarios. Dado a que últimamente la compañía ha sido involucrada en distintos actos de corrupción; ha decidido implementar un control mucho más estricto tanto para los pilotos, rutas, y autobuses, como para los usuarios, dicha solución constara de un sistema local y de un sistema web, con el objetivo de que la información generada en el sistema local sea de carácter público, con lo cual pretenden mitigar los señalamientos en contra de la empresa; para lo cual se podrá exportar cierta información del sistema local al sistema web, además el sistema será capaz de generar informacion que en el futuro los ayudara a tomar decisiones.

APLICACIÓN DE ESCRITORIO

Se desarrollara una aplicación de escritorio, en el lenguaje C y C++, utilizando el IDE QtCreator, la cual contendrá los siguientes módulos:

- Archivos (Cargar, y Guardar)
- Graficas
- Panel de Control (Crear, Ver, Actualizar, Eliminar)

ARCHIVOS

Se deberá de cargar archivos en formato 'csv' de memoria secundaria (disco duro), a estructuras dinámicas almacenadas en memoria primaria (RAM), para los siguientes registros:

- Autobuses
- Pilotos
- Estaciones
- Rutas
- Parqueos
- Usuarios
- Transacciones por Autobús

La estructura de los mismos se define a continuación:

Archivo de Autobuses:

Placa, Modelo, Estado

Archivo de Pilotos:

DPI, Nombre, Edad, Genero

Archivo de Estaciones:

Código, nombre, ubicación

Archivo(s) de Ruta:

Codigo_Estacion1, Codigo_Estacion2, … , Codigo_EstacionN

Archivo(s) de Parqueos:

Placa1, Placa2, Placa3, Placa4

Placa5, 0, Placa6, 0

0, Placa7, Placa8, Placa9

Archivo de Usuarios:

DPI, #_Tarjeta, Nombre, Edad

Archivo(s) De Transacciones:

DPI_1, DPI_2, … , DPI_N

AUTOBUSES: Tabla Hash

Cada autobús posee los siguientes atributos:

- Placa (String): Combinación de números y caracteres
- Modelo (String): Combinación de caracteres
- Estado (String): Combinación de caracteres

Los distintos autobuses serán almacenados en una tabla hash de tamaño m=37, se utilizara como llave k el valor ascii total (suma de ascii de c/carácter) de la placa del autobús, se utilizara como función de dispersión la función aritmética modular, y como política de resolución de colisiones dispersión abierta utilizando listas simplemente enlazadas

PILOTOS: Tabla Hash

Cada piloto posee los siguientes atributos:

- DPI (String): Combinación de números
- Nombre (String): Combinación de caracteres
- Edad (Int)
- Género (Char): Carácter único, ‘M’ y ‘F’

Los distintos pilotos serán almacenados en una tabla hash de tamaño m=37, se utilizara como llave k el valor ascii total (suma de ascii de c/carácter) del DPI del piloto, además se utilizara como función de dispersión la función aritmética modular, y como política de resolución de colisiones dispersión abierta utilizando un arboles AVL, y llave el nombre para los árboles, convirtiendo a ASCII el mismo

ESTACIONES: Lista Simplemente Enlazada

Cada estación posee los siguientes atributos:

- Código (String): Combinación de números y caracteres
- Nombre (String): Combinación de caracteres
- Ubicación (String): Combinación de caracteres

Las distintas estaciones serán almacenadas en una lista simplemente enlazada.

RUTAS: Lista Circular Doblemente Enlazada de Grafo(s)

Cada archivo de ruta poseerá en su primer línea la ruta de ida al destino, el primer registro será el origen, y en la segunda línea la ruta de regreso al origen, y será una combinación de N códigos de Estación, se podrán cargar distintos archivos de rutas, por lo que habrá una lista circular doblemente enlazada de rutas, cada nodo de esta lista poseerá un nodo al origen del grafo dirigido que representa la ruta, una estación puede estar tanto en la ruta de ida como en la ruta de regreso al origen

PARQUEOS: Lista Circular Doblemente Enlazada de Matrices Ortogonales

Cada línea del archivo de parqueos, representara una fila de la matriz, y cada elemento separado por ‘,’ (coma), representara una columna, y será una combinación de placas de autobuses; si el valor es 0, el espacio del parqueo está disponible (Se deberá de apreciar en la gráfica). Se podrán cargar distintos archivos de entrada por lo que se tendrá una lista circular doblemente enlazada, la cual poseerá en cada uno de sus nodos una referencia a la matriz de un parqueo

USUARIOS: Lista Simplemente Enlazada Mapeada Lexicográficamente

Cada usuario posee los siguientes atributos:

- DPI (String): Combinación de números
- Tarjeta (String): Combinación de caracteres
- Nombre (String): Combinación de caracteres
- Edad (Int)

Los distintos usuarios serán almacenados en una lista simplemente enlazada, que simulara una tabla hash, la cual poseerá un tamaño inicial de m=37, se utilizara como llave k el valor ascii total (suma de ascii de c/carácter) del DPI del usuario, que tendrá como límite el 80% de la capacidad de la estructura, para lo cual al llegar a este número, se procederá a realizar un ‘rehashing’ y expandirá en un 15% el tamaño actual utilizando el valor primo más cercano, con lo cual se podrá seguir insertando datos. Se utilizara como función de dispersión la función aritmética modular, y como política de resolución de colisiones dispersión cerrada, búsqueda lineal. Se podrá cargar distintos archivos, sin embargo el contenido de cada uno de ellos será agregado sobre la misma estructura

BLOQUE:

Se deberán poder crear distintos bloques, a dichos bloques se les podrá asignar los siguientes elementos:

- Piloto
- Autobús
- Rutas (3 – 5)
- Fecha

Cada ruta formara un nodo de una lista doblemente enlazada, y se podrá seleccionar cada una de estas en cualquier momento, el mínimo de rutas que tendrá el bloque es de 3 que son las obligadas a realizar por los pilotos, sin embargo estos pueden llegar a realizar 1 o 2 rutas extras en caso lo deseen, por lo que el máximo de rutas que podrán realizar es de 5

Para cada bloque se cargaran ya sea 3 o 5 archivos de transacciones por lo que cada archivo corresponderá a una ruta (en orden de carga), la estructura del mismo se describirá a continuación:

TRANSACCIONES: Árbol AVL

Cada archivo representara las distintas transacciones realizadas a lo largo de una ruta, y poseerá una serie de valores que representaran los DPI's correspondientes a los usuarios, cada línea de los archivos representara los distintos usuarios presentes en cada estación, a lo largo de una ruta (ida y vuelta), por lo que si una ruta pasa por 6 estaciones de ida al destino, y 4 estaciones de regreso al origen, existirán un total de 10 líneas en el archivo (si hay más líneas se ignoraran), por ende existirán 10 árboles AVL, uno perteneciente a cada estación, en el cual tendrá almacenado los distintos usuarios en esa estación.

Se debe de tomar en cuenta que se debe de poder visualizar el estado de usuarios en el autobús en cada estación, por lo que cada nodo de la ruta (grafo), debe de poseer una referencia al árbol AVL y su estado en cada estación

VER INFORMACION

Se debe de poder desplegar la información de todos los nodos de todas las estructuras mencionadas anteriormente, mediante el ingreso de la información necesaria para el despliegue de la misma

REPORTES: Árbol B[5]

Como se mencionó anteriormente la empresa CSSiga, pretende tomar decisiones en base a la información obtenida de las distintas transacciones del sistema, por lo que se deberán de generar los siguientes reportes:

- Usuarios por Bloque
- Usuarios por ruta

Se deberá de poder almacena los reportes en archivos, para lo cual se recorrerá in-orden el árbol y se almacenara dentro del archivo, además se deberá de almacenar información perteneciente al bloque/ruta que pertenecen

BITACORA DE TRANSACCIONES: Lista Simplemente Enlazada

Se deberá de almacenar en una lista simplemente enlazada, un log de transacciones, que contendrá toda la configuración inicial, del sistema, para que la configuración pueda persistir en disco, para lo cual cada nodo de esta estructura contendrá información necesaria de cada una de las transacciones.

ALMACENAMIENTO BITACORA DE TRANSACCIONES:

Como se mencionó anteriormente la bitácora de transacciones deberá persistir en disco, para lo cual dicha estructura será almacenada en un archivo con extensión '.#carnet-log' (ej. 201318613-log), y será almacenado encriptado utilizando el algoritmo de encriptación de preferencia por el estudiante (recomendados: DED, Cesar).

RECUPERACION DE BITACORA:

Se deberá de poder cargar los archivos creados en el inciso anterior, para lo cual se deberá de descifrar el contenido del archivo previamente encriptado, y luego crear la estructura, después de cargar la estructura se deberá de recrear las transacciones realizadas que generaron la bitácora

APLICACIÓN WEB

La aplicación web será desarrollada en el entorno Node.js, dicha aplicación consistirá en la carga de archivos al servidor, para lo cual existirán dos módulos.

CLIENTE: ADMINISTRADOR

Se deberá de tener una pantalla donde se acceda mediante las credenciales siguientes:

- Usuario: #carnet
- Contraseña: #carnet-edd-b-1s2018

(Ej. Usuario: 201318613 Contraseña: 201318613-edd-b-1s2018), en este módulo se podrá subir archivos de información correspondientes a los reportes bloque o ruta, para lo cual se almacenara en memoria en una estructura la cual posea como atributo adicional a la raíz del árbol el nombre del archivo desde el cual se carga la información, y se almacenara también en el servidor el archivo como tal.

CLIENTE: GENERAL

Este usuario podrá descargar los archivos de transacciones, así como visualizar los arboles correspondientes a los archivos subidos por el administrador.

GENERAL

Tanto en la aplicación de escritorio, como en la aplicación web existirán botones “Acerca De”, con los que se desplegara información acerca del curso, semestre, y datos siguientes del alumno:

- DPI
- Carnet
- Nombre
- Curso, Sección y Semestre

Observaciones

- Sistema Operativo: Linux
- IDE’s permitidos: QtCreator
- Las estructuras deben de ser realizadas por el estudiante, deben usar struct y punteros
- Herramienta de Graficas: Graphviz, y Viz(Node)
- Las GUI mostradas en este enunciado, son para orientar al estudiante sobre lo requerido, el estudiante debe de realizarlo como mejor le parezca

ENTREGABLES

- Código Fuente (Aplicación de Escritorio y aplicación Web)
- Ejecutable
- Manual Técnico
- Manual Usuario

FECHA Y MODO DE ENTREGA

- Martes 8 de Mayo de 2018 antes de las 23:59,entregas tarde tendrán automáticamente una nota igual a 0
- Se deben de enviar los entregables vía Dropbox, en un comprimido con el nombre: [EDD]Proyecto2_#carnet (e.g. [EDD]Proyecto2_201318613)
- Dropbox: https://www.dropbox.com/request/8i2ePWychND6rvYl8Q0s

Dudas y Consultas

- Se responderán dudas en horario de DSI o vía correo mediante el asunto
  "[EDD]Duda Proyecto 2"

Requerimientos Mínimos

- Visualización correcta de las estructuras mediante graficas

Penalizaciones

- Si ocurre un NULL Pointer Exception, la nota obtenida será 0 automáticamente
- Si se cierra la aplicación repentinamente, la nota obtenida será 0 automáticamente
- Las representaciones graficas de las estructuras son esenciales para verificar si se realizó de manera correcta la estructura, por lo que si no se representan de esta manera NO se calificara
- Si se detecta el uso de librerías como QStack, QQueue, QList, etc, para C/C++ la nota obtenida será 0 automáticamente
- Las representaciones graficas deberán de mostrarse integradas en las GUI, y realizadas utilizando Graphviz, de no ser así NO se calificara

***Copias de internet, así como copias totales o parciales, tendrán nota de 0 puntos, serán reportadas al catedrático y a la Escuela de Ciencias y Sistemas FI-USAC**