Universidad de Guadalajara

Reporte Solución Planteada

Garcia Ramírez Alberto

Problema Planteado:

Descripción: Una empresa de fabricación de equipos de cómputo necesita un programa que le ayude a llevar un control las llamadas que sus agentes han atendido durante el día. El programa deberá almacenar los nombres de los agentes con su respectivo horario, número de empleado, número de extensión, horas extras trabajadas, especialidad y la lista de usuarios atendidos durante su turno.

La secretaria encargada podrá agregar o eliminar agentes como sea necesario, así como usuarios atendidos por cada agente.

El programa contará con las funciones:

- Mostrar toda la lista de agentes almacenados, o bien filtrarlos por especialidad.
- Las especialidades para los agentes pueden ser: **Servidores, De escritorio, Portátiles, Linux, Impresoras, Redes**
- Agregar un agente
- Encontrar y mostrar los datos de un agente (número de empleado, nombre, especialidad, número de extensión, horario, horas extras, y usuarios atendidos) a través del nombre del agente y su horario
- Eliminar un agente a través de su nombre y horario
- Eliminar todos los agentes
- Ordenar agentes por nombre o por especialidad
- Agregar un cliente a la lista de un agente
- Eliminar un cliente de la lista de clientes atendidos por un agente
- Eliminar todos los clientes de la lista de un agente
- Modificar la duración la llamada de un cliente
- Guardar toda la lista de agentes y clientes atendidos en almacenamiento secundario
- Leer toda la lista agentes y clientes desde almacenamiento secundario

- Mostrar la lista de agentes con o sin lista de clientes

Características del programa:

- El paradigma de programación debe ser Orientado a Objetos
- El programa debe basarse en el uso de listas doblemente ligadas para los agentes, y listas simplemente ligadas para los usuarios atendidos por cada agente.
- El método de ordenamiento debe ser recursivo.
- Todos los TDA y sus funciones deben ser programadas por el alumno y contenerse en bibliotecas o colecciones bien definidas. Es decir que no habrán de utilizarse los TDA, o funciones para TDA incluidas en el lenguaje de programación elegido.

El programa debe hacerse en un lenguaje de programación portable

Para atender este problema voy a realizar un programa en lenguaje c++, el programa se ejecutara en la consola.

Hare diferentes clases para cada satisfacer las necesidades del programa.

"Nombre" para los atributos: nombre de pila y apellido, y los métodos para su manejo.

"Horario" para los atributos: hora y minuto, y los métodos para su manejo.

"Duracion" para los atributos: hora, minuto y segundo, y los métodos para su manejo.

"Cliente" para los atributos: nombre(Nombre), hora de llamada(Horario) y duración(Duracion) y los métodos para su manejo.

"Agente" para los atributos: nombre(Nombre), hora de entrada y hora de salida (Horario), numero de extensión, horas extras trabajadas y una especialidad, que puede ser entre Servidores, De escritorio, Portátiles, Linux, Impresoras, Redes, Lista de Clientes(ListadeClientes) y los métodos para su manejo.

"NodoCliente" con los atributos: siguiente(apuntador al siguiente nodo), cliente(Cliente), solo un apuntador porque la lista es simplemente ligada, y los métodos para su manejo.

"NodoAgente" con los atributos: siguiente(apuntador al siguiente nodo), anterior(apuntador al nodo anterior), Agente(Agente) y los métodos para su manejo.

"ListaCliente" con los atributos:

ultimoInsertado(apuntador al nodo que comienza la lista, “Ancla”),

primerInsertado(apuntador al nodo que termina la lista),

auxiliar(apuntador que va a ser usado repetidas veces para diferentes funciones como comenzar un recorrido de la lista o salvar la dirección para eliminar Nodos),

auxiliar2(apuntador que será usado para salvar una dirección de un nodo que se va a religar en la lista),

intercambiar(función que realiza intercambios de datos entre nodos pero solo se usa internamente por eso esta en los atributos),

copiar(función que realiza inserciones en la lista pero solo se usa internamente por eso esta en los atributos)

y las funciones públicas:

isEmpty (evalúa si la lista esta vacia y retorna un verdadero o un falso),

insertar(crea un nodo que se enlaza a la lista y se le agrega sus datos),

elimina(busca el nodo con los datos ingresados, lo elimina y religa la lista),

primerNodo(retorna la dirección del Nodo que inicia la lista),

ultimoNodo(retorna la dirección del nodo que finaliza la lista),

anterior(busca el nodo que antecede al nodo recibido y retorna esa dirección),

siguiente(retorna la dirección recibida en su atributo siguiente),

localiza(retorna la direccion del nodo después de buscarlo con los datos ingresados),

ordena(realiza el método iterativo quicksort con las direcciones del ultimoInsertado y el primerInsertado),

recupera(recibe un cliente y retorna una cadena con los datos del mismo),

toString(retorna una cadena con todos los datos de los clientes que existen en la lista, va nodo por nodo hasta llegar a la dirección “null”),

guardar al disco(crea un archivo e ingresa los datos de los clientes, uno por uno hasta llegar al ultimo nodo),

leer del disco(busca un archivo con el nombre de la lista, si no lo encuentra manda una excepción, si lo encuentra extrae los datos y los inserta a la lista actual),

eliminar todo(inicia en el primer nodo, guarda la dirección, avanza y la dirección guardad se elimina y toma la siguiente dirección hasta llegar a “null”).

“ListaAgentes” con los atributos:

ultimoInsertado(apuntador al nodo que comienza la lista, “Ancla”),

primerInsertado(apuntador al nodo que termina la lista),

auxiliar(apuntador que va a ser usado repetidas veces para diferentes funciones como comenzar un recorrido de la lista o salvar la dirección para eliminar Nodos),

auxiliar2(apuntador que será usado para salvar una dirección de un nodo que se va a religar en la lista),

intercambiar(función que realiza intercambios de datos entre nodos pero solo se usa internamente por eso esta en los atributos),

copiar(función que realiza inserciones en la lista pero solo se usa internamente por eso esta en los atributos)

y las funciones públicas:

isEmpty (evalúa si la lista esta vacia y retorna un verdadero o un falso),

insertar(crea un nodo que se enlaza a la lista y se le agrega sus datos),

elimina(busca el nodo con los datos ingresados, lo elimina y religa la lista),

primerNodo(retorna la dirección del Nodo que inicia la lista),

ultimoNodo(retorna la dirección del nodo que finaliza la lista),

anterior(retorna la dirección recibida en su atributo anterior),

siguiente(retorna la dirección recibida en su atributo siguiente),

localiza(retorna la direccion del nodo después de buscarlo con los datos ingresados),

ordenaPorNombre(realiza el método iterativo Quicksort con la condicion que se ordene en base al nombre y con las direcciones del ultimoInsertado y el primerInsertado),

ordenaPorEspecialidad(realiza el método iterativo Quicksort con la condicion que se ordene en base a la especialidad y con las direcciones del ultimoInsertado y el primerInsertado),

recupera(recibe un agente y retorna una cadena con los datos del mismo),

toString(retorna una cadena con todos los datos de los agentes que existen en la lista y sus listas de clientes, va nodo por nodo hasta llegar a la dirección “null”),

toStringAgente(retorna una cadena con todos los datos de los agentes que existen en la lista, va nodo por nodo hasta llegar a la dirección “null”),

toStringEspecialidad(retorna una cadena con todos los datos de los agentes que comparten la especialidad ingresada en la lista, va nodo por nodo hasta llegar a la dirección “null”),

guardar al disco(crea un archivo e ingresa los datos de los agentes, uno por uno hasta llegar al ultimo nodo),

leer del disco(busca un archivo con el nombre de la lista, si no lo encuentra manda una excepción, si lo encuentra extrae los datos y los inserta a la lista actual),

eliminar todo(inicia en el primer nodo, guarda la dirección, avanza y la dirección guardad se elimina y toma la siguiente dirección hasta llegar a “null”).

“MenuCliente” con los atributos:

MiLista(es un apuntador a una ListaCliente)

Y las funciones publicas:

menuPrincipal(ingresa a un menú que con cada opción puede mandar a submenús y cada opción llama a una función del menu)

agregarCliente(sin parámetros, llama la función insertar de la lista)

agregarCliente(con parámetros que recibe dos horarios,crea un cliente, llama la función inserta de la lista y manda es cliente para que se inserte)

borrarCliente(crea un cliente con los datos a buscar y los manda a la función eliminar y manda el cliente para que lo busque y elimine de la lista)

modificarCliente(crea un cliente con datos ingresados por el usuario, busca el cliente que coincida con los datos ingresados, se imprimen los datos y se pide ingresar la nueva duración con la que se guardara el cliente modificado)

ordenarLista(manda a llamar al método de ordenamiento en la lista)

mostrarLista(manda a llamar al método toString y después muestra los datos en pantalla)

guardarAlDisco(crea un archivo “Lista de Agentes.txt” que almacena la lista de agentes, pero al momento de guardar un agente, se manda guardar su lista de clientes con su “nombre(del agente).clientes.txt”, y asi sucesivamente hasta que termine con todos los agentes.)

leerDelDisco(Leer del disco, busca en la carpeta “Archivos” la lista de agentes y también busca la lista de clientes de cada agente, para al final quedar la lista de agentes y sus lista de clientes ya cargadas.)

borrarLista(manda a llamar la método de eliminar todo de la lista)

“MenuAgente” con los atributos:

listaDeAgentes(es un apuntador a una lista de Agentes)

menuClientes(es una lista de clientes)

Y las funciones publicas:

menuPrincipal(ingresa a un menú que con cada opción puede mandar a submenús y cada opción llama a una función del menu)

agregarAgente(sin parámetros, llama la función insertar de la lista)

borrarAgente(crea un agente con los datos a buscar y los manda a la función eliminar y manda el cliente para que lo busque y elimine de la lista, y elimine la lsita de clientes)

modificarAgente(crea un agente con datos ingresados por el usuario, busca el agente que coincida con los datos ingresados, se imprimen los datos y se pide ingresar los datos del agente para ser ingresados en el agente buscado)

modificarListaDeClientesPorAgente(se busca un agente y se manda a llamar al menú con la lista de clientes del agente, donde se puede agregar,modificar,etc.)

ordenarPorNombre(manda a llamar al método de ordenamiento por nombre en la lista)

ordenarPorEspecialidad(manda a llamar al método de ordenamiento por especialidad en la lista)

mostrarListaDeAgentes(manda a llamar al método toStringAgentes y después muestra los datos en pantalla)

mostrarListaDeClientesPorAgente(manda a llamar al método toString y después muestra los datos en pantalla)

mostrarListaDeClientesPorEspecialidad(manda a llamar al método toStringEspecialidad y después muestra los datos en pantalla)

mostrarAgente(busca con los datos ingresados el agente y muestra los datos completos de éste en pantalla)

guardarAlDisco(crea un archivo “Lista de Agentes.txt” que almacena la lista de agentes, pero al momento de guardar un agente, se manda guardar su lista de

clientes con su “nombre(del agente).clientes.txt”, y asi sucesivamente hasta que termine con todos los agentes.)

leerDelDisco(Leer del disco, busca en la carpeta “Archivos” la lista de agentes y también busca la lista de clientes de cada agente, para al final quedar la lista de agentes y sus lista de clientes ya cargadas.)

borrarLista(manda a llamar la método de eliminar todo de la lista)