UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE CIENCIAS Y SISTEMAS

LABORATORIO DE ESTRUCTURAS DE DATOS

SECCION C

PROYECTO FASE 1

MANUAL TECNICO

DENNIS MAURICIO CORADO MUÑÓZ

CARNET: 202010406

CUI: 3032329780108

GUATEMALA, GUATEMALA, 20/02/2022

Introducción

Las estructuras de datos es una forma en la que la información se puede organizar en una computadora para que puedan ser utilizados de manera eficiente. Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones y algunos son especializados para tareas específicas.

Durante el desarrollo del proyecto se hizo uso de estructuras lineales las cuales están comprendidas en listas, pilas, colas, listas circulares, entre otras. Esto con el objetivo de manejar la información de una manera dinámica y así poder brindar una solución más optima al problema planteado.

Índice

Objetivos

Objetivos Generales

- Poder aplicar todos los conocimientos adquiridos durante en el desarrollo del curso y el laboratorio de estructuras de datos, y así poder crear una aplicación que sustente los requerimientos propuestos y manejar la información provisionada de forma óptima.

Objetivos Específicos

- Hacer uso de estructuras lineales aprendidas durante el curso y ponerlas en práctica en la aplicación de simulación.
- Utilizar el lenguaje de programación Java para implementar estructuras de datos lineales.
- Hacer uso de la herramienta Graphviz para poder graficar los datos contenidos dentro de las estructuras lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación.

Requisitos del Sistema

Para el correcto funcionamiento de la aplicación se deben tener en cuenta algunos requisitos mínimos bastante específicos, dentro de los cuales tenemos:

- Procesador – Intel Pentium o superior.
- RAM – 1GB de RAM o superior.
- Espacio en Disco – 1GB libres en disco.
- Sistema Operativo – Windows, MacOS o Linux.
- Tener instalado el software de Java JDK versión 16 o superior.
- Tener instalado el software de Graphviz versión 2.48 o superior.

Estos requisitos garantizan el poder utilizar la aplicación de una manera correcta.

Lógica del Programa

Para.

Clases Principales

Clase ‘Proyecto_fase1’

Método main

Dentro de este método inicial únicamente se instancia lo que es el menú para poder dale inicio al programa y poder ejecutar todo lo necesario.

Clase ‘Menú’

Dentro de esta clase se hace como su nombre lo indica el menú y a su vez se instancia la clase administrador desde la cual se manejará las diferentes funcionalidades del programa.

Constructor

Dentro del constructor de la clase podemos encontrar una llamada al método menú para desplegar así las diferentes opciones de la aplicación.

Método menú

En este método se crean lo que son los diferentes menús que contiene la aplicación, se inicia con el menú principal que tiene las opciones de parámetros iniciales, ejecutar paso, estado en memoria de las estructuras, reportes, acerca de, salir. En base a la opción elegida por el usuario se crea un switch-case.

En el caso de ser parámetros iniciales se despliega un submenú que contiene las opciones de carga masiva, cantidad de ventanillas y una opción para regresar al menú inicial, de igual forma dentro de este case se crea otro switch-case, que dependiendo de la opción se sabe si se hará uso de algunos de los métodos dentro del administrador.

En el caso de ser realizar paso hace una llamada a un método del administrador al igual que el caso 3, en el caso 4 se despliega otro submenú con su switch-case para cada opción del menú, y de esa forma se trabaja dentro del método menú.

Clase ‘Administrador’

Dentro de esta clase se realizan instancias a las otras diferentes clases, como la cola de recepción, las ventanillas, los atendidos, entre otros.

Método ‘read_json’

Dentro de este método se hace uso de la librería ‘json_simple’ para realizar la lectura del archivo json así como se envía la información de los diferentes clientes contenidos dentro de este archivo a la cola de recepción.

Método ‘create_ventanilla’

Se apoya de un ciclo para crear en base al numero de ventanillas elegido por el usuario las diferentes ventanillas y envía la información al objeto ‘ventanillas’ de tipo ListVentanilla.

Método ‘realizar_paso’

Este método utiliza bastante de los próximos métodos a describir para segmentar el código y poder realizar un correcto mapeo de errores en caso los hallan, dentro de este método se llaman métodos para crear usuarios random, sacar clientes de lista de espera, entregar imágenes a clientes en espera, enviar clientes de las ventanillas a la lista de espera, verifica que se encuentre alguna ventanilla vacía y en base a eso saca un usuario de la cola de recepción y lo envía a ser atendido a ventanilla.

Método ‘agregar_usuario’

En este método se saca un cliente de la cola de recepción, se verifica que este cliente no sea nulo, y en caso de no serlo se agrega a la ventanilla que este disponible en ese momento.

Método ‘enviando_a_espera’

Este método verifica por medio de un método interno de la lista de ventanillas si se puede sacar algún cliente de la ventanilla en caso de que este ya haya entregado todas sus imágenes, de ser así, que este usuario se puede sacar, esta verificación devuelve otra lista de clientes ya que en caso se encuentren mas clientes que ya hayan dado sus imágenes en el mismo momento entonces esta lista de clientes se envía a la lista de espera y se cargan sus imágenes a cada impresora. Así como se libera el espacio que estos ocupaban en cada ventanilla.

Método ‘entregando_imagenes’

Como bien su nombre lo indica en este método se hace lo que es la entrega de las diferentes imágenes a cada usuario que se encuentra en la lista de espera, en caso que se espere una imagen a color se hace una consulta a esta impresora si ya la puede devolver dado que esta impresora tarda 2 pasos en entregar una imagen. En el caso que se espere una imagen a color esta se entrega de forma inmediata a su usuario correspondiente.

Método ‘sacando_de_espera’

Acá se saca a un cliente de la lista de espera en caso ya haya recibido todas sus imágenes, esto se puede corroborar dado que desde el momento que se forman en la cola de recepción se le adjudica un valor a una variable propia que indica la cantidad de imágenes que estos llevan. En el momento que salen de la lista de espera se envían a una lista de clientes atendidos.

Método ‘create_random’

Durante este método se crea los que son los clientes e imágenes de cada uno de forma random, se hace uso de 2 arreglos, uno para los nombres y otro para los apellidos y se asocian estos de manera random para crear así el nombre de los nuevos clientes y también se crea de manera random la cantidad de imágenes a color y a blanco y negro, posteriormente se envían a la cola de recepción

Método ‘estructuras_memoria’

En este método se hace uso de diferentes métodos internos de cada una de las diferentes estructuras de datos que se están usando durante la ejecución de este programa, los métodos internos de estas estructuras generar lo que es una imagen por medio de lo que es la herramienta de graficar ‘graphviz’, para poder ver así de manera grafica el estado de cada una de las estructuras.

Método ‘generar_graphviz_imgs’

Dado que hay algunas estructuras como las colas de impresión que trabajan por separado, cada estructura es independiente y por facilidad de ver ambas estructuras en un mismo archivo, se crea este método dentro de este método se hace un recorrido de los diferentes nodos que contiene cada estructura y se añaden a una cadena en especifico desde la cual se escribirá el respectivo archivo ‘.dot’ para poder así obtener la imagen del estado de cada cola de impresión.

Método ‘clientes_mas_pasos’

En este método y los posteriores se hace uso de una estructura denominada reportes dentro de la cual se generan lo que son los distintos reportes que soporta la aplicación, en este método se hace lo que es un ordenamiento para obtener el usuario con mas pasos en el sistema.

Método ‘top5_mas_color’

Al igual que el anterior se hace uso de un método de la clase lista reportes desde la cual se ordenan los nodos de mayor a menor para obtener los 5 usuarios con mayor cantidad de imágenes a color.

Metodo ‘top5_menos_bn’

Se ordena de menor a mayor a los usuarios atendidos para obtener los 5 con menor cantidad de imágenes a blanco y negro.

Metodo ‘datos cliente’

Obtiene los datos de un cliente en específico haciendo una búsqueda por medio de los id de los diferentes clientes atendidos.

Clases de las Estructuras de Datos

Clase ‘ColaRecepcion’

Clase Node

La clase nodo dentro de esta otra clase es utilizado para crear la estructura de datos, dentro de esta clase encontramos información que sirve enormemente al desarrollo del proyecto dado que acá se crea lo que es el nodo que contiene la información del cliente y diferentes variables que cambian su valor a lo largo del proyecto.

Metodo push

En este método se realiza lo que es la inserción de clientes a la cola de recepción con sus respectivos datos.

Metodo pop

En este se sacan de la cola los clientes que están próximos a ser atendidos y se realiza el cambio de posición de los diferentes nodos.

Metodo get_index

Este método únicamente devuelve la cantidad de clientes que han ingresado a la cola.

Metodo generate_graphviz

Este método se encarga de generar la vista grafica de la estructura, realiza ciclos para obtener la información y la concatena en una cadena que al final se creara un archivo de extensión dot sobre el cual se debe ejecutar un comando para así obtener la imagen png de esta estructura.

Metodo print_clientes

Se realiza un recorrido por los distintos nodos y se imprime su información pertinente en consola.

Clase ‘ListVentanilla’

Clase Node

Dentro de esta clase se crean los nodos usados para la lista de ventanillas, estos nodos contendrán variables como el numera de la ventanilla, su estado para saber si esta ocupada o disponible, un nodo de la cola de recepción el cual seria el cliente, la pila de imágenes que van recibir del cliente y su siguiente.

Metodo append

Dentro de este método se crean nuevos nodos en base a la cantidad de ventanillas que el usuario ingreso.

Metodo find_space

Este verifica que la ventanilla tenga su estado como disponible para así poder pasar a un cliente de la cola de recepción.

Metodo free_space

Este método se encarga de liberar el espacio en una ventanilla, realiza una búsqueda por las diferentes ventanillas para saber a cuál corresponde el id brindado y a esta ventanilla le regresa sus valores como el cliente, el estado y la pila de imágenes a un estado inicial.

Metodo se_puede_sacar

Este método verifica que un cliente pueda ser extraído de la ventanilla, se verifica si la cantidad de imágenes blanco y negro y a color son 0, esto quiere decir que ya entrego todas sus imágenes la ventanilla y se saca de la ventanilla para poder ser almacenado en la lista de espera.

Metodo pop_cliente

Este método se utiliza de manera interna dentro de la lista, dado que se realiza una búsqueda de los diferentes nodos y se verifica cual es el nodo que se puede sacar, una vez encontrado se devuelve el nodo.

Metodo set_user

Como su nombre lo indica este método se encarga de asignar un cliente a la ventanilla, así como también cambia el estado de la ventanilla para hacer saber que se encuentra ocupada e incrementa un paso en el usuario para no perder la cantidad de pasos que este tardara en el sistema.

Metodo set_image

En este método se realiza lo que son las entregas de las imágenes de los diferentes clientes que se encuentran siendo atendidos, se envía una imagen a la pila de imágenes que tiene cada ventanilla.

Metodo get_ventanilla

Este método devuelve el numero de ventanilla realizando una búsqueda en base al id del cliente que se solicita.

Metodo generate_graphviz

Este método se encarga de generar la vista grafica de la estructura, realiza ciclos para obtener la información y la concatena en una cadena que al final se creara un archivo de extensión dot sobre el cual se debe ejecutar un comando para así obtener la imagen png de esta estructura.

Metodo print_list

Imprime la información que contienen las ventanillas.

Clase ‘CircularClientesEspera’

Clase Node

Dentro de esta clase se crean lo que son las variables necesarias para el funcionamiento de la cola circular, se define una variable del tipo del nodo de la

cola de recepción para poder almacenar los clientes que lleguen de la ventanilla, así como una lista de imágenes que va a corresponder a cada cliente, y las ubicaciones de siguiente y anterior.

Meted append

Dentro de este método se crean nuevos nodos y se conectan de manera bidireccional manteniendo así el comportamiento de la lista circular doblemente enlazada.

Metodo set_images

Este método realiza una búsqueda del cliente dentro de la lista para poder entregarle una imagen que ha sido impresa por las colas de impresión.

Metodo finalizado

Verifica si un cliente ya recibió todas las imágenes que llevaba, de ser así devuelve el nodo de la lista para realizar las operaciones pertinentes con él.

Metodo delete_node

Este método lo que realiza es una verificación por medio de la dirección de memoria proporcionada y al encontrar esta dirección eliminada al nodo de la lista, y restablece los nodos siguientes y anterior.

Metodo increment_paso

Incrementa un paso de toda la lista de clientes que están en espera.

Metodo generate_graphviz

Este método se encarga de generar la vista grafica de la estructura, realiza ciclos para obtener la información y la concatena en una cadena que al final se creara un archivo de extensión dot sobre el cual se debe ejecutar un comando para así obtener la imagen png de esta estructura.

Metodo print_list

Imprime la lista de clientes en espera y su respectiva cantidad de imágenes.

Clase ‘QueueBwPrinter’

Esta clase se encarga de almacenar las imágenes a blanco y negro enviadas desde la ventanilla, como su nombre lo indica es una cola y va despachando una imagen por cada paso realizado en el sistema.

Clase ‘QueueColorPrinter’

Similar a la cola anteriormente descrita, esta se encarga de llevar las imágenes a color que son trasladadas desde las ventanillas, la principal diferencia de esta cola es que despacha las imágenes cada 2 pasos y para eso hace una validación ya que lleva el conteo de pasos y si se aplica el modulo de 2 a la cantidad de pasos se puede obtener si ya termino la impresión de las imágenes.

Clase ‘ListaAtendidos’

Esta clase se encarga de almacenar a los clientes que terminaron su proceso por la aplicación, son almacenados y únicamente pueden ser consultados para los reportes.

Clase ‘ListaImagenes’

Clase Node

En la clase nodo de la lista de imágenes únicamente se almacena el tipo de imagen que es.

Metodo append

Este método permite la creación de nuevos nodos con su tipo de imagen cada uno.

Metodo get_index

Devuelve la cantidad de imágenes que han sido ingresadas a la lista.

Metodo is_empty

Verifica si la lista está vacía.

Metodo print_list

Imprime la lista de imágenes con su tipo.

Clase ‘PilaImagenes’

Clase Node

La clase nodo de la pila de imágenes contiene información que relaciona las imágenes con el cliente y cada nodo tiene información para saber si es una imagen a blanco y negro o a color.

Metodo push

Crea nodos que son almacenados en la lista, considerando que debe obtener los valores booleanos de las imágenes para saber que tipo son y el id del cliente del que vienen dichas imágenes.

Metodo pop

Saca las imágenes de la pila, esencialmente este nodo se usa para la entrega de imágenes de la ventanilla a las impresoras.

Metodo cant_imagenes

Devuelve la cantidad de imágenes que ha recibido del cliente que esta siendo atendido en la ventanilla.

Metodo get_nodo

Devuelve el nodo inicial de la pila para su debido uso.

Metodo is_empty

Verifica que la pila este vacía o no.

Metodo print_images

Imprime en consola las imágenes, con el id del cliente y permitiendo saber si es a blanco y negro o a color.

Clase ‘ColaApoyo’

Como su nombre lo indica esta es una cola que esta siendo utilizada a manera de apoyo para otras estructuras, almacena objetos de tipo de nodo de la cola de recepción.

Clase ‘ListaReportes’

Esta lista en especial es utilizada para los reportes que pueden ser generados por la aplicación, se recurrió utilizar esta estructura dinámica debido a que los reportes requieren algoritmos de ordenamiento que pueden cambiar la estructura central de los clientes atendidos, por ende al implementar esta estructura no se tiene ningún efecto sobre las demás y es capaz de devolver los reportes solicitados, en esta se realiza bastante repetición de código debido a que los ordenamientos son parecidos sin embargo cada uno necesita un atributo muy específico.

Conclusiones

El correcto uso de estructuras dinámicas permite realizar diferentes actividades haciendo uso de pocos recursos en el sistema, cada una de las diferentes estructuras lineales permiten lo que es una amplia gama de opciones que se pueden realizar, así como el hecho de que las estructuras sean programadas por uno mismo permite implementar métodos propios que faciliten la ejecución o desarrollo de programa, así como algoritmos de búsqueda y ordenamiento dentro de estas mismas estructuras