

**Estructuras de Datos**

**Instituto Tecnológico de Costa Rica**

**I Semestre 2019**

**TAREA PROGRAMADA 3**

**Integrantes:**

* **Danny Piedra Acuña 2016210168**
* **Esteban Jiménez Valverde 2017112650**

**Fecha de Entrega: Jueves 20 de Junio, 2019**

**ÍNDICE**

Portada\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Índice \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

Introducción\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

Descripción de Clases\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5

Interfaz Gráfica\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

Resultados Obtenidos\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

**INTRODUCCIÓN**

El manejo y la estructuración de la información corresponde a un concepto fundamental en las Ciencias de Computación. El cómo se maneja y gestionan los recursos de memoria tiene una serie de implicaciones a gran escala delegando consecuencias a otros factores de mucha importancia como lo son la construcción de sistemas de información versátiles, dinámicos y funcionales. El procesamiento y organización en memoria de grandes cantidades de datos, así como la construcción de más modernas y estratégicas soluciones a problemas que surgen diariamente, haciendo el uso más optimizado de la memoria puesto que este es un recurso finito que tienen los computadores.

Bajo este concepto, a través de las décadas han surgido innumerables tipos de algoritmos apoyados por una múltiple variedad de lenguajes de programación, unos más complejos que otros.

Han surgido muchas estructuras de datos desde las más simples e intuitivas como la organización de forma secuencial o en bloques de un tipo de dato como las listas enlazadas que almacenan un dato y una referencia al siguiente, por lo que tanto la obtención y almacenamiento debe ser secuencial, esto implica que se debe recorrer cada uno de los elementos hasta llegar al lugar buscado, esto no parece muy eficiente cuando contamos con una alta cantidad de registros.

Algunos otros utilizan alguna clase de abstracción del mundo real para concretar las definiciones de estos algoritmos como es el caso de los árboles binarios de búsqueda.

Y otros más complejos nos permiten dar solución a problemas de mucha importancia, como lo que en matemática se llama Teoría de Grafos.

Estos nos permiten dar solución a problemas de distribución y acceso de elementos conectados de alguna manera, esto tiene una amplia cantidad de aplicaciones en el mundo real así como en problemas propios de la Computación.

Cabe resaltar que una vez que se construyen estos algoritmos que nos solucionan varios problemas como la manera en que almacenamos la información, la obtenemos y ordenamos. Surgen así mismo nuevos problemas dados en función de la entrada de información que le damos al algoritmo.

Así que se han originado una serie de algoritmos “básicos” de Grafos que nos permiten dar soluciones interesantes a preguntas como. ¿Cuál es el camino más óptimo para accesar a todos los nodos pertenecientes al grafo?, o ¿De que manera podemos procesar todos los nodos involucrados en el Grafo para visitarlos a profundidad o en amplitud (por niveles)? así como problemas más clásicos tales como ¿Cuál es el camino más corto partiendo de un nodo específico hacia otro? O ¿Cuál es el camino más corto para accesar a todos los otros nodos de la forma más óptima partiendo de un nodo específico?

Así que en el desarrollo de este Proyecto Programado se estará utilizando un algoritmo de grafos que nos permiten dar solución a un problema real como lo es la Simulación de un Sistema de Transporte, el cuál llamaremos “MIWA”. Adicionado a este, se estará utilizando algoritmos de Colas con Prioridades. Así como Estructuras de Datos para Almacenamiento de Archivos externos como Árboles B y Archivos Directos preservados en disco.

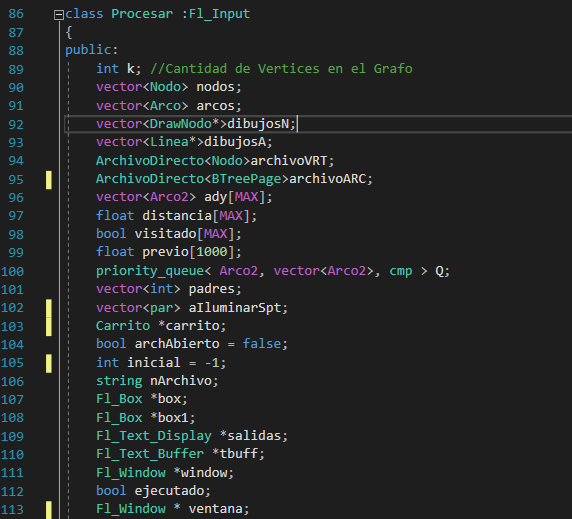
El propósito de este proyecto por ende es el de hacer una aplicación de Simulación Vial funcional que de solución a temáticas citadas previamente como encontrar el camino más corto a todos los nodos en una Red o trazar la ruta más barata a otro componente de la red utilizando uno de los algoritmos más fundamentales de los grafos. El algorítmo Dijsktra.

**Descripción de Clases**

Primeramente, es menester comentar la estrategia que se utiliza en esta tarea para dar solución a la construcción de este Sistema de Simulación de Transporte. Se ha creado una clase maestra o principal llamada “Procesar”, esta se considera el cerebro de la aplicación puesto que se encarga interpretar y tokenizar las instrucciones ingresadas por el usuario, además de implementar la instanciación de los componentes de la Interfaz Gráfica. Esta también hará uso de otras clases complementarias que se estarán explicando en brevedad.

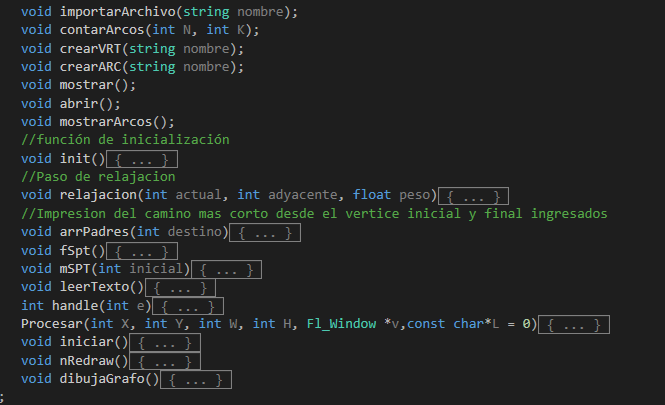
La clase Procesar es significativamente la más compleja puesto que conserva un conjunto de atributos propios de la librería FLTK, vectores que encapsulan otras clases y funciones que se encargan de crear los archivos de acceso directo y las páginas del Árbol B.

La primera parte de la clase es la siguiente:



Tal como puede observar, la Clase cuenta con cuadros de texto, cuadros de despliegue, vector de Nodos de Árboles B. Así como el conjunto de vectores para los padres que utilizará el algoritmo de Dijkstra.

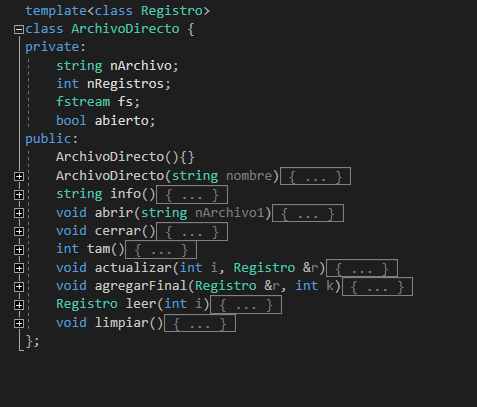
Adicionalmente se agregan los métodos propios de Procesar que permiten la ejecución del algoritmo:



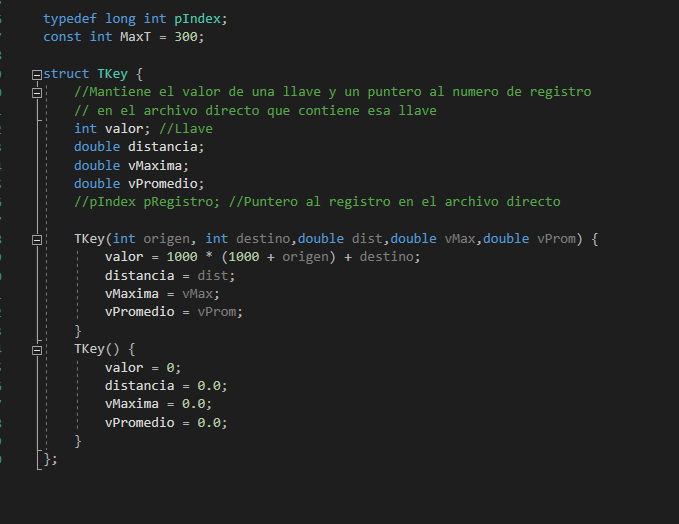
De manera que el ciclo de ejecución del programa consiste en que el constructor llame instancia el constructor por defecto de la clase Fl\_Input y posteriormente llamará al método iniciar, éste se encargará de inicializar los componentes de la Interfaz Gráfica asignando los parámetros establecidos previamente. Seguidamente, la función int handle() se encargará de poner a la espera que la tecla enter se presione para detonar un evento y llamar a la función leeTexto() que se encargará de tokenizar la entrada del la cinta llamada salidas y se encargará de llamar a cada función respectivamente.

Por otra parte tenemos un conjunto de clases complementarias que se utilizaron para almacenar toda la información en disco y en memoria dinámica también.

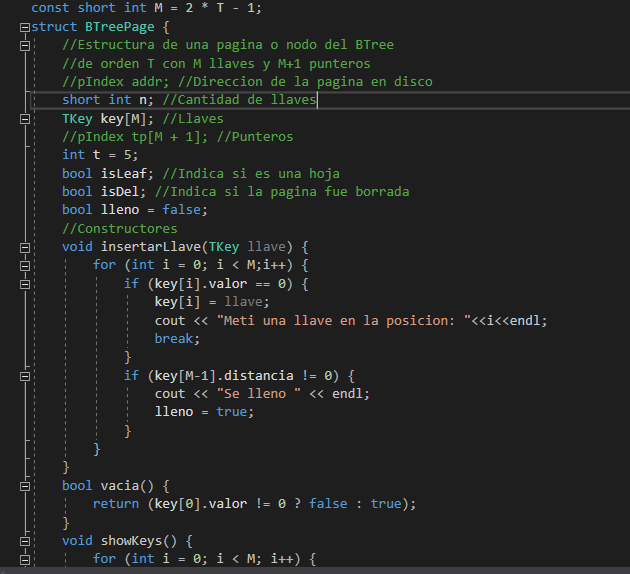
Esta clase, nos permite tener un archivo de escritura en disco, así como una escritura de la cabecera del archivo, puesto que es una clase con template nos permite escribir cualquier tipo de objeto en el archivo directo.



Posteriormente, contamos con una clase muy trivial TKey que nos permite encapsular una llave, así como instanciar una llave con la formula facilitada por los requerimientos del proyecto.



Por otro lado, tenemos la clase BTreePage que nos permite almacenar un conjunto de llaves almacenadas en un arreglo que nos permite insertar llaves para el almacenamiento en disco de cada una de las páginas del Árbol B que se crean almacenando la información de los arcos del Grafo para poder acceder a ella de manera más sencilla.



**Interfaz Gráfica**

Si bien la interfaz gráfica es estéticamente muy simple y arbitraria según los requerimientos de la tarea, se encontró un grado de dificultad al trabajar con eventos gestionados al presionar teclas, y posteriormente encontrar una librería que permitiera dibujar sobre el canvas de manera flexible y customizable así como compatible con los eventos de teclas.

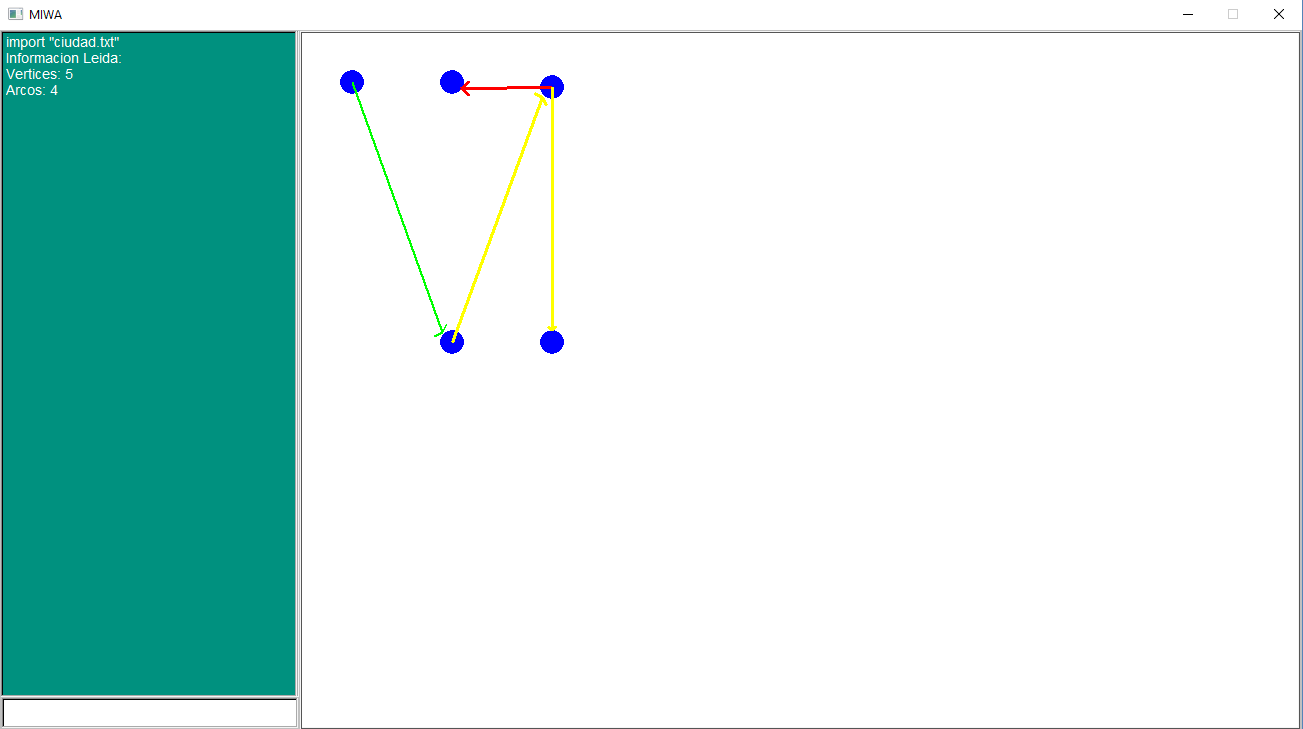
Sin embargo, después de una profunda investigación encontramos que la librería FLTK iba a solventar los requerimientos de la tarea así como facilitar el cumplimiento de los requerimientos adicionales como la animación de figuras a través del canvas.

La interfaz luce de la siguiente manera:



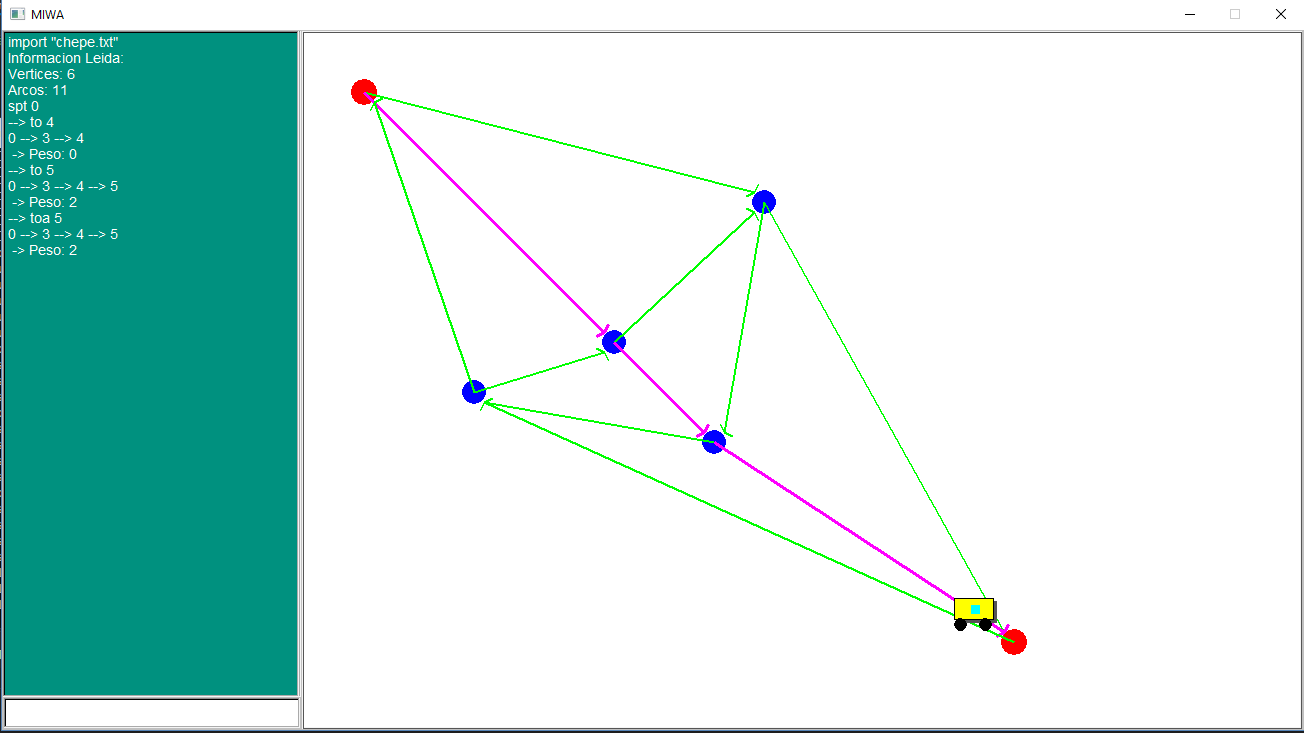
Como puede observar la ventana solamente contiene un espacio para el ingreso de las instrucciones, la cinta histórica de instrucciones y donde muestra el resultado de aplicar los algoritmos y demás mensajes adicionales como errores, sugerencias, etc. Y finalmente el canvas donde se dibujan los nodos y se procesan las animaciones.

Adicionalmente, posterior a hacer la inclusión de un grafo a la interfaz gráfica esta se mostrará de la siguiente manera:



Los nodos se cargan a la interfaz gráfica con los colores de sus arcos en función de su velocidad promedio. Se plantea dibujar los nodos pequeños para que estos puedan encajar a la perfección puesto que la aplicación acepta un máximo de 1000 nodos posibles a ser cargados de disco a la aplicación.

Finalmente, posterior a la inserción del archivo, a calcular el camino más corto de un nodo arbitrario, y finalmente la ejecución de la animación de ese camino calculado la interfaz gráfica luciría de la siguiente manera:



**Resultados obtenidos**

Finalmente, en cuanto a la lista de resultados obtenidos llegamos a la conclusión que la aplicación cumple con los requerimientos establecidos por el documento. Se generaron pruebas con múltiples archivos, y se aplicaron validaciones sutiles para dotar de robustez al programa.

Se listan los objetivos cumplidos de la tarea:

* **Se implementó el algoritmo de Dijkstra.**
* **Se implementó una interfaz gráfica capaz de mostrar una red archivada en disco.**
* **Se diseñaron las clases necesarias para la abstracción del problema y el encapsulamiento de múltiples fuentes de información a través de Patrón de Diseño Modelo Vista Controlador (MCV).**
* **Se implementaron Archivos Directos y Árboles B persistidos en disco.**
* **Se implementaron las instrucciones *import “nombre.txt”, node x, arcs x, clear, close, cleart, open “nombre”, spt v, to w, toa w*.**
* **Se aplicaron validaciones al programa.**