*Instituto Tecnológico de Costa Rica*

*Unidad de Computación*

*[****“Proyecto Sistema de Mapas”****]*

*Esteban Blanco*

*Carlos Villafuerte*

*Sede San Carlos*

*[13/11/2015]*

**Introducción:**

Se presenta un proyecto con la ideología de un GPS o mapa, pero utilizando estructuras de tipo grafo en el lenguaje de c++, este proyecto no tendrá tanto potencial como el localizador global, pero se podrán cargar las ciudades desde un archivo y agregar nuevas ciudades al mismo archivo. Se realizaran recorridos en diferentes formas y consultas dependiendo de las necesidades del usuario. Se alimentaran archivos con la información de choferes los cuales poseerán una ruta o rutas determinadas este se cargara de forma dinámica en la ejecución del programa. Dependiendo de cuando sea llamado.

La inserción modificación o eliminación de ciudades se harán en el menú de ciudades respectivo, ahí se podrá almacenar un nuevo conductor o una ruta a un conductor determinado. La inserción y cierre de vías se harán en el menú caminos, solo tendrá esas dos opciones, para las consultas habrá un respectivo menú que poseerá todas las posibles consultas que un usuario podría hacer o requerir los reportes igual que los otros poseerá un menú, en el cual se indican las posibilidades de este.

La información se cargada en el menú de archivos, cabe descartar que el programa una vez compilado no poseerá información hasta que la cargue en este menú. Después de realizar ciertas consultas es mejor que presiones la opción 10 del menú de consultas para poder seguir haciendo consultas ya que el grafo se ensuciara y dejara de recorrer ciertos caminos que pueden estar cerrado en ese momento. Los recorridos se harán de forma dinámica una vez escogidos no se detendrán hasta terminar, cabe la posibilidad que algún recorrido se en cicle por ello deberá de ejecutar nuevamente el programa esto se debe a lo ya dicho antes sobre la limpieza del grafo…

El programa tiene la opción de salir de cualquiera de los menús y regresar al menú principal, este es el que dirige todas las operaciones,.

**Análisis del problema:**

Se necesita implementar un programa en c++ que recorra un mapa de ciudades el cual estará grabado o ejecutando desde una estructura de tipo grafo, para almacenar las ciudades de dicho mapa.

Este poseerá varios reportes entre los cuales se destacan tendrá la inserción eliminación y modificación de tanto de ciudades como caminos, averiguar si existe una ruta que puede llegar de un lugar X a un lugar Y, pasando por cierta cantidad de ciudades, también se tendrá que consultar cual es la ruta más corta o rápida dependiendo de lo que se ocupe el algoritmo tendrá que pasar por cada camino buscado las rutas que no estén cerradas en ese momento, por otro lado se asignará a cada ciudad un color diferente al de su ciudad vecina para poderlo distinguir, pero usando la cantidad mínima de colores, además se llevará un registro del uso de las rutas de los conductores, con el fin de determinar las rutas más congestionadas o más usadas por los usuarios

El programa deberá brindar información al usuario final, en el caso de que desee visitar todas las ciudades en un solo recorrido tendrá que poseer un menú para el personal de mantenimiento de caminos, en el caso que quiera recorrer todos los caminos, en un solo recorrido, para ver su estado.

Dicha información deberá ser almacenada en archivos. Al iniciar el programa debe existir la opción para cargar los datos desde archivo en el caso que el usuario así lo desee, una vez cargados los datos el usuario puede ingresar más datos o modificarlos. Cuando el programa finaliza, se deberá almacenar toda la información ya modificada por el usuario en los respectivos archivos.

**Solución del Problema:**

La inserción, modificación y eliminación de ciudades y caminos se harán en funciones respectivas, las cuales le pedirán lo datos a los usuarios de mantenimiento, cada función velara por que no se repitan datos en el sistema de lo contrario tirara un mensaje de que la estructura esta repetida y no la insertara esto con el fin de evitar inconvenientes a la hora de realizar un reporte o consulta los caminos serán independiente de cada ciudad, habrá una opción de imprimir las ciudades ya se en ampliación y profundidad.

* Estructura usada para almacenar las ciudades.

**struct Ciudade{**

**string nombreCiudad;**

**string colorCiudad;**

**bool visitado;**

**struct Ciudades \* sigCiudad;**

**struct Caminos \* subListaCaminos;**

**Ciudades(string nombreLugar)**

**{**

**nombreCiudad = nombreLugar;**

**colorCiudad = "";**

**visitado = false;//false;**

**sigCiudad = NULL;**

**subListaCaminos = NULL;**

**}**

**}\*mapaCiudades;**

* Estructura usada para almacenar los caminos de cada ciudad

**struct Caminos{**

**int distanciaKm;**

**int velocidad;**

**bool dirigido;**

**bool paso;**

**bool cerrado;**

**struct Ciudades \* CiudadDestino;**

**struct Caminos \* sigCamino;**

**Caminos(int distancia, int kph, bool siesdirigido, struct Ciudades \* destino)**

**{**

**distanciaKm = distancia;**

**velocidad = kph;**

**dirigido = siesdirigido;**

**sigCamino = NULL;**

**CiudadDestino = destino;**

**paso = 0;**

**cerrado = 0;**

**}**

**}**

Se poseerán archivos respectivos para almacenar la información estos serán consultados en el menú de archivos que se encuentra en el principal, de aquí se cargaran los datos al sistema, muy importante una vez que ya a subidos tales datos no se puede volver a elegir la opción ya que esta incurriría en un error de duplicación.

* Estructuras de los archivos para las consultas.

struct rutasArchivo{

char origen[20];

char destino[20];

int distancia;

int velocidad;

bool dirigido;

}registroCaminos;

struct archivoConductores{

int id;

char nombreChofer[15];

char tipoVeiculo[15];

long enlaceRutas;

long izq;

long der;

}registroConductores;

struct ciudad{

char nombre[10];

};

struct archivoRutasC{

struct ciudad lugares[10];

long sig;

}registroRutasC;

Para averiguar si la ciudad des conexa o no implementara un algoritmo que recorrerá cada ciudad y buscara si de la ciudad actual se puede llegar a todas por uno o varios caminos, de lo contario la función retorna un 0 como false y se imprimirá que nos es conexa.

La impresión de una sola ciudad en el recorrido se hace con el ciclo de Eulen almacenando las ciudades que se tocan una sola vez en una lista, el recorrido es recursivo, si llegara al destino y las ciudades visitas exceden en uno a las ciudades del grafo se puo hacer el recorrido he imprimirá esa lista de lo contrario no se podrá hacer y mostrara un mensaje de error.

La impresión de recorrer el grafo con un solo camino conlleva las misma lógica que el anterior nada más que este se modifica la variable bolean que se encuentra en el camino para solo pasar una vez por él, es recursivo y preguntara si ya paso por ese camino o no, las caminos libres los guardara en una lista para imprimirla después.

La consulta de los colores se realiza con el teorema de colores, el cual expresa “Cualquier mapa es posible pintarlos con un máximo de 4 colores”, con esto se genera el algoritmo que busca una ciudad y pregunta si este apunta a otra ciudad si con el mismo color o a ella la apuntan con ese mismo color si no lo hace, la pintara del color en el que pregunto. El mapa será pintado solo con los colore necesarios o los que ocupe.

Para imprimir el camino más corto se tomara en cuenta la distancias la menor sumas de ellas será el recorrido más corto, esto se almacenara en otra lista, el las almacenara dependiendo de donde este como es recursivo eliminara las ciudades que no le competen.

De otra forma el de la ruta más rápida se tomaran las velocidades o rutas con mayor velocidad y sus kilómetros una vez obtenido esto se dividen entre ellos y le enseñara al usuario el tiempo que puede durar.

La consulta de las rutas alternas va de la mano con lo de el algoritmo de cierre de camino, esta tomara el camino a averiguar si está cerrado o no he imprimirá las rutas con las que se pueden llegar a un destino sin pasar por el camino.

Imprimir la cantidad de usuarios que viajan hacia una ciudad, esta consulta el archivo de rutas conductor y averigua cuántos de ellos llegan al mismo destino, en rutas diferentes.

Y la última consulta le pedirá al usuario el id del conductor y este recorrerá todas las rutas de dicho conductor, he imprimirá en una lista.

Para los reportes se consultan los archivos en el sistema he imprimen todo el contenido que tenga en ese momento.

**Etapas:**

1 .Avance

**struct Caminos{**

**int distanciaKm;**

**int velocidad;**

**bool dirigido;**

**bool paso;**

**bool cerrado;**

**struct Ciudades \* CiudadDestino;**

**struct Caminos \* sigCamino;**

**Caminos(int distancia, int kph, bool siesdirigido, struct Ciudades \* destino)**

**{**

**distanciaKm = distancia;**

**velocidad = kph;**

**dirigido = siesdirigido;**

**sigCamino = NULL;**

**CiudadDestino = destino;**

**paso = 0;**

**cerrado = 0;**

**}**

**}**

struct rutasArchivo{

char origen[20];

char destino[20];

int distancia;

int velocidad;

bool dirigido;

}registroCaminos;

struct archivoConductores{

int id;

char nombreChofer[15];

char tipoVeiculo[15];

long enlaceRutas;

long izq;

long der;

}registroConductores;

struct ciudad{

char nombre[10];

};

struct archivoRutasC{

struct ciudad lugares[10];

long sig;

}registroRutasC;

2. Avance

void InsertarCiudad(string nombreLugar)

{

/\*\* \brief Inserta la ciudades nuevas a la lista mapasCiudades..

\*

\* \param nombreLugar: es el nombre del nuevo lugar por crear, si ya existe no lo agregar

\* \return NULL

\*

\*/

struct Ciudades \* nnC = new Ciudades(nombreLugar);

nnC->sigCiudad = mapaCiudades;

mapaCiudades = nnC;

}

void InsertarDatosCiudad()

{

/\*\* \brief Funcion que pide al usuario el nombre de la nueva ciudad y llama a la funcion para insertarla

en la lista de ciudades..

\*

\* \param NULL

\* \return NULL

\*

\*/

string nombreLugar;

string colorLugar;

cout<<"\n\tDigite el nombre de la ciudad Nueva"<<endl;

cout<<"\*\*\*";

cin>>nombreLugar;

if(buscar(nombreLugar) != NULL){

cout<<"\n\t\t\*\*\*\*\*\*\*\*\*\*La ciudad con ese nombre ya existe..\*\*\*\*\*\*\*\*\*\*\n"<<endl;

return;

}

else

{

InsertarCiudad(nombreLugar);

cout<<"\n\t\t\*\*\*\*\*\*\*\*\*\*Se agrego con exito la ciuadad..\*\*\*\*\*\*\*\*\*\*\n"<<endl;

return;

}

}

void VerCiudades()

{

/\*\* \brief Imprime todas las ciudades que se encuentra en el mapa con sus respectivos destinos

\*

\* \param NULL

\* \param NULL

\* \return NULL

\*

\*/

int contador = 1;

struct Ciudades \* tempCiudades = mapaCiudades;

struct Caminos \* tempCaminos;

cout<<"\n\tLa ciudades en el mapa son:\n "<<endl;

while(tempCiudades != NULL)

{

cout<<"\t "<<contador<<" "<<tempCiudades->nombreCiudad<<endl;

tempCaminos = tempCiudades->subListaCaminos;

cout<<"\t\t Sus rutas son: "<<endl;

while(tempCaminos != NULL)

{

cout<<"\t\t\t "<<tempCiudades->nombreCiudad<<" a "<<tempCaminos->CiudadDestino->nombreCiudad<<endl;

tempCaminos = tempCaminos->sigCamino;

}

contador++;

tempCiudades = tempCiudades->sigCiudad;

}

cout<<"\n\n";

}

void BorrarCiudad()

{

/\*\* \brief Borra la ciudad deseada, primero borra los caminos que vallas hacia ella y luego la borra

de la lista de ciudades...

\*

\* \param

\* \param

\* \return

\*

\*/

string nombreCiudad;

cout<<"Digite el nombre de la Ciudad a borrar"<<endl;

cout<<"\*\*\*";

cin>>nombreCiudad;

struct Ciudades \* ciudadEncontrada = buscar(nombreCiudad);

if(ciudadEncontrada == NULL)

cout<<"Ciudad no Existe"<<endl;

else

{

struct Ciudades \* tempCiudades = mapaCiudades;

struct Caminos \* tempCaminoAnterior;///Guarda el camino antes de el que se va a eliminado

///Borra las rutas que van hacia la ciudad que se quiere borrar...

while(tempCiudades != NULL)

{

if (tempCiudades->subListaCaminos == NULL){

continue;

}

if(tempCiudades->subListaCaminos->CiudadDestino->nombreCiudad == nombreCiudad)

{

tempCaminoAnterior = tempCiudades->subListaCaminos->sigCamino;

delete tempCiudades->subListaCaminos;

tempCiudades->subListaCaminos = tempCaminoAnterior;

}

else

{

struct Caminos \* tempCamino = tempCiudades->subListaCaminos;

tempCaminoAnterior = tempCamino;

while(tempCamino != NULL)

{

if(tempCamino->CiudadDestino->nombreCiudad == nombreCiudad)

{

tempCaminoAnterior->sigCamino = tempCamino->sigCamino;

delete tempCamino;

}

tempCaminoAnterior = tempCamino;

tempCamino = tempCamino->sigCamino;

}

}

tempCiudades = tempCiudades->sigCiudad;

}

///Borra la ciudad de la lista Ciudades

tempCiudades = mapaCiudades;

struct Ciudades \* ciudadAnterior;

if(mapaCiudades->nombreCiudad == nombreCiudad)

{

ciudadAnterior = mapaCiudades->sigCiudad;

delete mapaCiudades;

mapaCiudades = ciudadAnterior;

}

else

{

ciudadAnterior = mapaCiudades;

while(tempCiudades != NULL)

{

if(tempCiudades->nombreCiudad == nombreCiudad)

{

if(tempCiudades->sigCiudad == NULL)

{

ciudadAnterior->sigCiudad = NULL;

delete tempCiudades;

}

ciudadAnterior->sigCiudad = tempCiudades->sigCiudad;

delete tempCiudades;

}

ciudadAnterior = tempCiudades;

tempCiudades = tempCiudades->sigCiudad;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Código de Caminos \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InsertarCaminos(struct Ciudades \* cOrigen, struct Ciudades \* cDestino, int distancia, int velocidad,bool dirigido)

{

/\*\*\brief Inserta los nuevos caminos de una ciudad a otra.. Si no se encuentra, no agregara nada

\*

\* \param cudadOrigen: el nombre de la ciudad de Origen;

ciudadDestino: el nombre de la ciudad hacia donde el camino va;

distancia: kilometros que posee esa carretera

velociadad: la velocidad a la que puede ir un automovil.

\* \return NULL

\*

\*/

if(cOrigen == NULL || cDestino == NULL){

cout<<"\n\t\t\*\*\*\*\*\*\*\*\*\*Datos Inexistentes..\*\*\*\*\*\*\*\*\*\*"<<endl;

return;

}

else{

struct Caminos \* nnCamino = new Caminos(distancia,velocidad,dirigido,cDestino);

nnCamino->sigCamino = cOrigen->subListaCaminos;

cOrigen->subListaCaminos = nnCamino;

cout<<"\n\t\*\*\*\*Se inserto con exito\*\*\*\*"<<endl;

}

}

void InsertarDatosCaminos()

{

/\*\* \brief Inserta los datos de los caminos, si el camino origen tiene el mismo color que el camino destino no se

inserta.....

\*

\* \param

\* \param

\* \return

\*

\*/

string ciudadDestino;

string ciudadOrigen;

int distancia;

int velocidad;

string dirigido;

cout<<"Digite la ciudad Origen\n\*\*\*";

cin>>ciudadOrigen;

cout<<"Digite la ciudad Destino\n\*\*\*";

cin>>ciudadDestino;

struct Ciudades \* COrigen = buscar(ciudadOrigen);

struct Ciudades \* CDestino = buscar(ciudadDestino);

cout<<"Digite la velocidad en k/h permitida\n\*\*\*";

cin>>velocidad;

cout<<"Digite la distancia\n\*\*\*";

cin>>distancia;

cout<<"Digite si el camino es Ida y Vuelta\n(si/no)\n\*\*\*";

cin>>dirigido;

if(dirigido == "si")

{

InsertarCaminos(CDestino,COrigen,distancia,velocidad, true);

InsertarCaminos(COrigen,CDestino,distancia,velocidad,true);

}

else

{

InsertarCaminos(COrigen,CDestino,distancia,velocidad,false);

}

}

void VerCaminosCiudad(struct Ciudades \* inicio)

{

/\*\* \brief Imprime los caminos de una ciudad ingresada por el usuario, hacias los destinos que este tenga...

\*

\* \param estructura del camino buscado

\* \param

\* \return

\*

\*/

if(inicio == NULL)

return;

struct Caminos \* tempCaminos= inicio->subListaCaminos;

if (tempCaminos == NULL){

cout<<"Esta ciudad no posee caminos"<<endl;

return;

}

while(tempCaminos != NULL)

{

cout<<"\t\t"<<inicio->nombreCiudad<<" a "<<tempCaminos->CiudadDestino->nombreCiudad<<endl;

tempCaminos = tempCaminos->sigCamino;

}

cout<<"\n\n";

}

void AuxVerCaminosCiudad()

{

/\*\* \brief Es la funcion que ayuda a la busquedad de un camino y sus rutas

\*

\* \param

\* \param

\* \return

\*

\*/

string nombreCiudad;

cout<<"Digite el nombre de la Ciudad"<<endl;

cin>>nombreCiudad;

struct Ciudades \* temp = buscar(nombreCiudad);

if(temp == NULL)

cout<<"Ciudad no Existe"<<endl;

else

VerCaminosCiudad(temp);

}

//Borra un camino entre dos ciudades. Las ciudades se reciben en una funcion aparte para la recepcion de datos y utilizan la funcion buscar

void BorrarCamino(struct Ciudades \* tempO,struct Ciudades \* tempD){

if (tempO == NULL or tempD == NULL){

cout<<"Datos Incorrectos"<<endl;

return;

}

else{

if (tempO->subListaCaminos == NULL){

cout<<"La Ciudad origen no tiene caminos...";

return;

}

else if(tempO->subListaCaminos->CiudadDestino->nombreCiudad == tempD->nombreCiudad){

struct Caminos \* tempCamino = tempO->subListaCaminos;

tempO->subListaCaminos = tempO->subListaCaminos->sigCamino;

tempCamino->sigCamino = NULL;

delete tempCamino;

}

else{

struct Caminos \* tempCamino = tempO-> subListaCaminos;

struct Caminos \* tempAnterior = tempO-> subListaCaminos;

while (tempCamino != NULL){

if (tempCamino-> CiudadDestino->nombreCiudad == tempD->nombreCiudad){

tempAnterior->sigCamino = tempCamino->sigCamino;

tempCamino->sigCamino = NULL;

delete tempCamino;

}

tempAnterior = tempCamino;

tempCamino = tempCamino->sigCamino;

}

}

}

}

void ModificarCamino(struct Ciudades \* tempO,struct Ciudades \* tempD){

if (tempO == NULL or tempD == NULL){

cout<<"Datos Incorrectos"<<endl;

return;

}

else{

if (tempO->subListaCaminos == NULL){

cout<<"La Ciudad origen no tiene caminos...";

return;

}

else{

struct Caminos \* tempCamino = tempO-> subListaCaminos;

while (tempCamino!= NULL){

if(tempCamino->CiudadDestino->nombreCiudad == tempD->nombreCiudad){

int nVelocidad;

int nDistancia;

cout<<"Digite la nueva distancia a asignar"<<endl;

cin>>nDistancia;

cout<<"Digite la nueva velocidad en k/h a asignar"<<endl;

cin>>nVelocidad;

tempCamino->distanciaKm = nDistancia;

tempCamino->velocidad = nVelocidad;

cout<<"\*\*\*\* Modificado con exito \*\*\*\*"<<endl;

return;

}

tempCamino = tempCamino->sigCamino;

}

}

}

}

**Análisis de resultado:**

|  |  |  |
| --- | --- | --- |
| *Tarea/Requerimiento* | Estado | *Observaciones* |
| Insertar y borrar ciudades, en cualquier momento del programa. | Completado | *Por defecto todos se insertan desde el archivo correspondiente y se guardan el él de forma dinámica* |
| Insertar, modificar y borrar caminos. | Completado |  |
| Insertar Conductores. | Completado |  |
| Insertar rutas. | Completado |  |
| Determinar si el grafo de transporte es conexo: si existe un arco entre cualquier par de vértices. O que todos los nodos se conecten entre sí... | Completado | *El grafo que se encuentra en el archivo no es conexo para realizar la consulta 2 y 3 se tiene que agregar vías.* |
| Imprimir el recorrido del ciclo que recorra todos los vértices del grafo pasando una sola vez por cada vértice. Imprimir el nombre de las ciudades respectivas. | Completado |  |
| Imprimir el recorrido del ciclo de todos los arcos del grafo pasando una sola vez por cada arco. | Completado | *Se tiene que hacer primero esta consulta y luego se agregan los caminos para hacerlos Conexo por la forma en que esta él mapa, se en cicla en la Ciudad Florencia \_ San Carlos* |
| Imprima la coloración de cada vértice del grafo, el algoritmo debe usar la menor cantidad de colores, como máximo cuatro colores, los vértice adyacentes tienen que tener un color distinto. | Completado |  |
| Imprimir la ruta corta (distancia) las ciudades X y Y. | Completado |  |
| Imprimir el tiempo más corto que tardaría trasladándose de una ciudad X a la ciudad Y. | Completado | *El tiempo capturado no se imprime por un problema de c++, no reconoce los float en la pantalla. Pero si se hace(a veces lo imprime)* |
| Mostrar todas las rutas alternas de comunicación entre la ciudad X y la ciudad Y en caso del cierre de un camino (arco). | Completado | *Tiene que ir a la función para cerrar las vías y luego entrar en esta.* |
| Imprimir la cantidad de usuarios que viajan hacia una ciudad X (destino). | Completado |  |
| Imprimir todas las rutas que realiza el usuario X. El id lo debe buscar en el archivo ordenado, después dirigirse al archivo de rutas realizando acceso directo. | Completado |  |
| Imprimir Archivo Conductores en formato de árbol binario, tal como se ve en el diagrama anterior. | Completado |  |
| Imprimir el Archivo Rutas, tal y como se ve en el diagrama anterior del archivo de rutas. | Completado |  |
| Realice un árbol en memoria principal de recorridos a partir de una ciudad X e imprimir las rutas con su respectivo uso. El recorrido para la impresión debe ser en “pre orden”. | Incompleto |  |

**Conclusiones:**

La compresión de la estructura tipo grafo genera confusión al principio y es un poco tediosa comprenderla, pero es la base para realizar todo el proyecto.

La sintaxis de los archivos es muy complicada, se debe tener una mejor percepción y entendimiento de los arboles binarios para poder realizar los recorridos en este.

**Recomendaciones:**

Dedicar más tiempo a la compresión y recorridos de grafos, buscar información acerca de los ciclos de Euler y el teorema de colores.

Pasar tiempo recorriendo archivos y comprendiendo la sintaxis de este para realizar los diferentes recorridos y generar la suficiente capacidad de saber que es lo que está haciendo.