UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

Rodriguez Rentería Jesus Alejandro

Código:215510307

Materia: Seminario de solución de problemas de estructuras de datos II

Profesor: Gómez Anaya David Alejandro

Sección: D16

Sede: Centro Universitario de Ciencias Exactas e Ingenierías (CUCEI)

Fecha:15 de noviembre del 2020

1. Planteamiento del problema.

Diseñar un mapa con el uso de grafos, se creará un grafo principal de implementación dinámica, la cual servirá de molde para la eventual creación de mas mapas. Se deben diseñar las funciones que permitan hacer movimientos de los personajes dentro del mapa.

2. Objetivos

Lograr la creación de un grafo dinámico

Diseñar el algoritmo para desplazamientos en un grafo dinámico.

Diseñar e implementar los métodos para poder almacenar los grafos en una lista y poder manipular sus atributos.

3: Marco teórico

Para esta actividad Se utiliza listas dinámicas y grafos dinámicos

Listas dinámicas en c++: Es un conjunto de nodos que se enlazan, su tamaño puede crecer mediante el tiempo de ejecución apéndice 1.

Grafos: Un grafo es una composición de un conjunto de objetos conocidos como nodos que se relacionan con otros nodos a través de un conjunto de conexiones conocidas como aristas. Apéndice 2.

Los grafos permiten estudiar las relaciones que existen entre unidades que interactúan con otras.

Podemos representar diversas situaciones o elementos con grafos.

4. Desarrollo

El código cuenta con 6 clases las cuales son: Clase Menu, Clase Edge, clase Vertex, Clase DynamicGraph, Clase Nodo, Clase Lista.

Clase menú: solo cuenta con elemento de impresión de datos no tiene atributos.

Clase Edge: Se encarga del almacenamiento de los datos de la arista como puede ser vértice origen y vértice destino, así como la ponderación de cada uno. Cuanta con los siguientes métodos:

- Edge(Vertex* destination, int weight):Constructor con parámetros
- string toString(): impresión de las conexines que generan las aristas con ponderación
- Edge* get_nextEdge(): En caso de que un vértice tenga múltiples aristas estas se almacena similar a una lista dinamica
- Vertex* get_destination():retorna el nombre del vertice al que apunta la arista.
- string recuperacadena():recupera en un solo string todos lugares de destino de un vertice

- char recuperades():regresa solo el destino de una arista

Clase Vertex: Se encarga de guardar el nombre del vertice, cuenta con un apuntador al siguiente vertice apuntadores a donde se almacenan sus aristas y un contador de las mismas. Sus métodos son:

- Vertex(char data): constructor con parametros
- void addEdge(Vertex* edge, int weight): agregar una aritas al vertice
- string toString(): impresión de los vértices
- void removeEdge(Vertex* v_e):eliminación de un vértice
- Edge* findEdge(Vertex* v_e):buscar un vértice
- Vertex *get_siguiente(): obtener el vértice siguiente
- char get_data(): retorna el nombre del vértice
- int get_EdgesCount();retorna el contador de aristas del vértice
- Edge* get_firtsEdge();obtenemos la primera arista del vértice

Clase DynamicGraph: Se encarga de guardar la posición del personaje en el mapa, el nombre del mapa, cuenta con conexiones hacia el primer vértice del grafo y un contador de vértices. Sus métodos son:

- void addVertex(char data): construcor con parametros
- void addEdge(char data_1, char data_2, int weight)agregar una arista
- string toString(); impresión del grafo global
- string toString2(); imprime partes del grafo individuales
- Vertex* findVertex(char data);busca un vertice por su nombre
- void removeVertex(char e); remover vertices
- Vertex* previousVertex(Vertex* v); obtener el vertice anterior
- void removeEdge(char data_1, char data_2); quitar una arista
- int getVertexCount() const; obtener el contador de vertices
- int getWeightEdge(char data_1, char data_2);obtener la ponderacion
- void addEdge2(char data_1, char data_2, int weight);
- string toFile();acomodar los grafos para su almacenamiento secundario
- void load();cargar los grafos para su ejecución
- char get_nombre(); obtener el nombre del grafo
- void set_visitados(string visi_);ingresar los lugares visitados
- void set_actual(char);ingresar posición actual en el grafo
- char get_actual(); obtener la posición del personaje en le grafo
- string lugares_adyacentes(); obtener los lugares adyacentes a la posición del personaje
- void despalzarse(char);método para hacer desplazamientos

Clase Nodo: Se encarga de el acomodo de la lista dinámica en la cual se almacenan todas las direcciones del grafo. Sus métodos son:

- Nodo();: construcotr sin parametros
- Nodo(DynamicGraph *, Nodo *);Ingresa apuntadores de un grafo y el nodo siguiente
- void set_graph(DynamicGraph *);ingreso la dirección de memoria de un grafo
- void set_siguiente(Nodo *);ingresamos el nodo siguiente

- Nodo *get_siguiente();obtenemos el nodo siguiente Nodo *get_anterior();
- DynamicGraph *get_graph();obtenemos la dirección del grafo almacenado
- void set_aterior(Nodo *);obtenemos el nodo anterior (no se usa en el código)

Clase Lista, cuenta con un solo atributo que es el ancla de la lista. Sus métodos son:

Lista();

void ingresa(DynamicGraph *)Ingresamos la dirección de memora de un grafo ya existente a nuestra lista

void toFile();Junto con el metdo tofile de DynamicGraph se encragan de guardar los datos en el archivo de texto

void take_of();Lectrura del archivo de texto y carga los registros de los grafos en la lista

bool mapa_repetido(char); buscamos el nombre del mapa en busca de alguna concidencia (es para validaciones)

void mostrar_mapa_principal(); llamamos los métodos para la impresión del grafo global

void find_grafo_principal();buscamos el grafo global en la lista (generalmente esta al inicio

void quitar_lista(char);quitamos un grafo de la lista

void find_grafo(char);buscar un grafo por nombre.

El programa inicia su ejecuin en el archivo main en el cual se despliegan las opciones a elegir, las cuales son

1. Crear grafo
2. Eliminar grafo

3 mostrar grafo

4. Acceder.

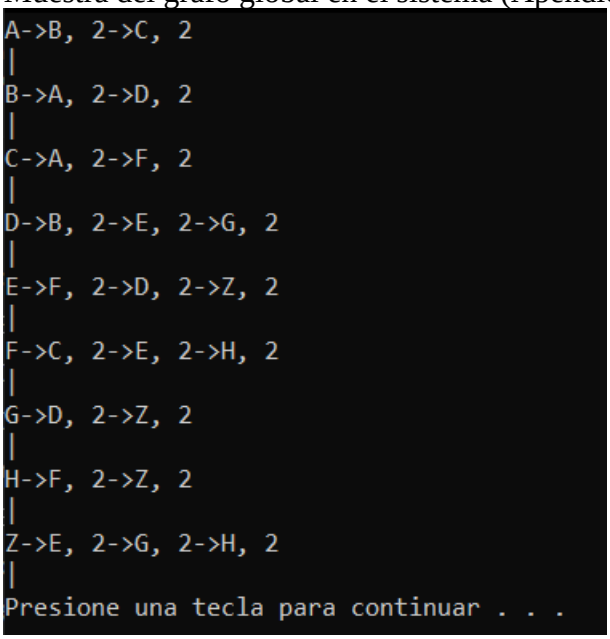
Dependiendo de la opción a seleccionar se realizan las acciones correspondientes a la frase.

5.Pruebas y resultados

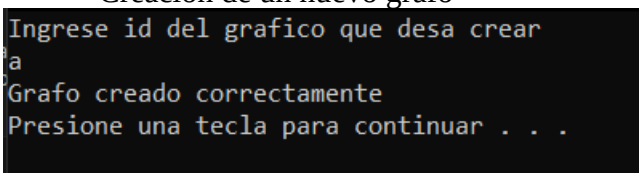
El menú principal y sus opciones

```
>>>>>>>>>> Menu principal <<<<<<<<
1. Creara grafo
2. Eliminar grafo
3. Mostrar grafo global
4. Acceder
5. Salir
```

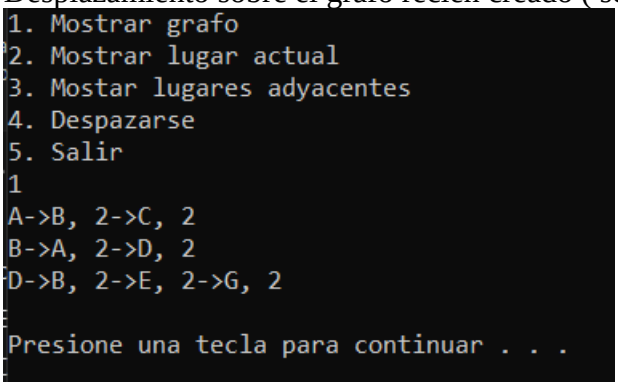
Muestra del grafo global en el sistema (Apendice2 muestra de manera visual)



Creación de un nuevo grafo



Desplazamiento sobre el grafo recién creado (se realizaron dos desplazamientos)

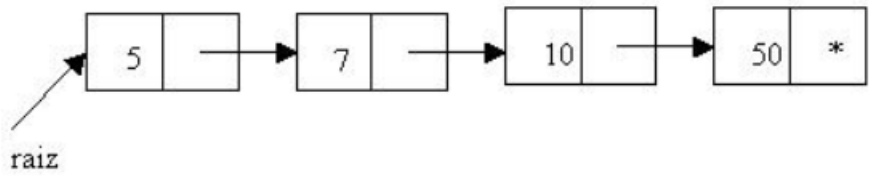


6. Conclusiones

La creación de grafos de manera dinámica forzar al programador a utilizar muchos punteros en el caso del leguaje de c++. Una cosa de la cual me petate fue que todas las clases correspondiente a el almacenamiento de los datos del grafo son friend class, lo cual da a entender que para la ejecución y operación las tres clases en ocasiones necesitan acceder a sus atributos y la de sus amigos. Esto puede acarrear algunos problemas de comprensión como fue en mi caso. Parte de esa complejidad se nota cuando, se necesita organizar los elementos para su almacenamiento secundario. En conclusión, es una estructura bastante compleja que requiere el majeo de muchas clases de manera simultanea para su correcta ejecución.

7. Apéndices

Apéndice 1.
Ejemplo de manera visual de como funciona una lista de manera dinámica en una computadora



Apéndice 2
La imagen que se presenta es complementaria al grafo global mostrado en pruebas y resultados es el mismo grafo solo que representado de manera más entendible en esta imagen