Tarea N°2
Programación en C: Aplicación de Grafos

Fecha límite de envío: domingo 23 de octubre, 23:55
Modalidad: trabajo en grupos de a lo más **dos** personas

I. Objetivos

El objetivo de este laboratorio es evaluar tu capacidad para:

1. Llevar a cabo un programa completo en el lenguaje de programación C, utilizando todos los elementos básicos que provee el lenguaje, sus tipos de datos básicos y extendiendo dichos elementos para ajustarse a la necesidad del problema.
2. Aplicar las buenas prácticas de programación para obtener un código legible y mantenible (orden, comentarios, identificadores representativos).
3. Ser capaz de cumplir con los requerimientos de una interfaz con exigencias específicas.
4. Diseñar e implementar estructuras que permitan trabajar con el TDA grafo y sus operaciones de búsquedas de camino.

II. Enunciado

El Metro de Santiago es uno de los sistemas de ferrocarril metropolitano más modernos de América Latina y el segundo más extenso, solo por detrás del de Ciudad de México, contando con 118 estaciones (y siete más en construcción actualmente, además de tres nuevas líneas) y una extensión de 140 km en siete líneas. Debido a su complejidad y la cantidad de gente que se transporta (con un promedio de alrededor de 2.6 millones de pasajeros al día), es conveniente conocer las rutas posibles entre estaciones y así calcular las mejores vías disponibles para desplazarse por la ciudad.

Para este problema, se tiene un archivo de texto plano con información sobre las estaciones, como se muestra en el código 1. El archivo completo se entregará como archivo adicional. El formato es el siguiente: la primera línea señala el total de estaciones. A continuación, vienen los nombres de cada una de las estaciones (tantas como la cantidad señalada en la primera línea, los puntos suspensivos representan la continuación de la lista). Finalmente vienen las conexiones entre estaciones, una por línea, siguiendo el formato `estación_1,estación_2,distancia`, donde la distancia está en metros.

Lo que se requiere es conocer el camino entre dos estaciones para los siguientes casos:

1. Se requiere conocer la **distancia total** entre dos estaciones de interés, por el camino más corto, en kilómetros.
2. Se requiere conocer cuáles estaciones hay que recorrer por el camino más corto.
3. Si se corta un tramo entre dos estaciones, se requiere conocer la ruta alternativa más corta (siempre por el metro) que permita llegar a destino (nota: ni el origen ni el destino de este requerimiento estarán en el tramo cortado).

Para lograr esto, se pide construir un programa en C que lea desde el archivo entregado las estaciones del metro y las conexiones entre estas y muestre un menú que permita elegir el caso a evaluar. Para los primeros dos casos, el programa pide el nombre de la estación de origen y el nombre de la estación de destino y, en el tercero, adicionalmente los nombres de las estaciones donde se corta el tramo (note que, si se corta un tramo, se cortan las conexiones entre dos pares de estaciones). Un ejemplo de la interfaz se presenta en los códigos 2, 3 y 4.

Código 1: Ejemplo de archivo de entrada

```
126
San Pablo
Neptuno
Pajaritos
Las Rejas
...
San Pablo,Neptuno,690
San Pablo,Pudahuel,1580
San Pablo,Lo Prado,570
Neptuno,San Pablo,690
Neptuno,Pajaritos,1030
Pajaritos,Neptuno,1030
Pajaritos,Las Rejas,860
Las Rejas,Pajaritos,860
Las Rejas,Ecuador,520
...
```

Código 2: Ejemplo de interfaz para opción 1

```
Seleccione opción:
1) Distancia entre estaciones.
2) Ruta entre estaciones.
3) Tramo alternativo en caso de corte.
>> 1
Estación de origen: República
Estación de destino: Santa Ana
Distancia mínima: 1.48 km
```

Así debe ser el menú:

```
Seleccione opción:
1) Distancia entre estaciones.
2) Ruta entre estaciones.
3) Tramo alternativo en caso de corte.
4) Salir
>>
```

Código 3: Ejemplo de interfaz para opción 2

```
Seleccione opción:
1) Distancia entre estaciones.
2) Ruta entre estaciones.
3) Tramo alternativo en caso de corte.
>> 2
Estación de origen: República
Estación de destino: Santa Ana
Ruta: República, Los Héroes, Santa Ana
```

Código 4: Ejemplo de interfaz para opción 3

```
Seleccione opción:
1) Distancia entre estaciones.
2) Ruta entre estaciones.
3) Tramo alternativo en caso de corte.
>> 3
Estación de origen: Los Héroes
Estación de destino: Baquedano
Conexión cortada 1: Moneda, Universidad de Chile
Conexión cortada 2: Santa Lucía, Universidad Católica
Ruta alternativa: Los Héroes, Santa Ana, Plaza de Armas, Bellas Artes, Baquedano
```

Para el tercer caso, considere que si no hay una ruta posible (por ejemplo, si se corta el tramo desde Trinidad hasta Elisa Correa y se quiere llegar hasta la Plaza de Puente Alto), el programa debe señalarlo.

III. Consideraciones en la Revisión

1. El tramado de la red debe ser modelado como un grafo, cualquier otro tipo de solución será considerada inválida y, por lo tanto, calificada con nota 1,0.
2. El programa debe seguir exactamente las reglas e interfaces señaladas en este documento.
3. El código debe seguir los principios del diseño funcional, es decir, todas las operaciones deben ser modularizadas y cada operación debe tener su propia función.
4. El código debe ser modular, es decir, cada grupo de operaciones debe estar declarada en su propio encabezado y definidas en su propio archivo de código fuente.
5. Las funciones deben tener, *como mínimo*, una o dos líneas describiendo su propósito, sus entradas y la salida que entregan. Adicionalmente, cualquier fragmento de código cuya lógica le parezca necesario explicar, por la razón que sea, debería estar apropiadamente comentado. Buenos comentarios son parte de las buenas prácticas y tanto este como este enlace contienen algunas guías de estilo generales que conviene tener en cuenta. También puedes revisar este enlace, en castellano.
6. Él código debe estar adecuadamente indentado, penalizándose códigos desordenados en este aspecto. Esto significa cosas como las llaves de apertura y cierre deben estar adecuadamente alineadas, todas las líneas dentro de un bloque deben estar al mismo nivel e indentadas con respecto a la cabeza del bloque que las contiene, etc.
7. El código no puede presentar más de una línea en blanco seguida, salvo entre funciones, donde debe haber dos entre el cierre de la función o instrucción previa y el inicio de los comentarios de la siguiente. Esta separación ayuda a la legibilidad, por lo que es conveniente usar una línea en blanco para separar partes o aspectos de la función.
8. Puedes trabajar con la IDE que más te acomode, no obstante, el código debiera poder ser compilado y ejecutado sin problemas en cualquier sistema que soporte las bibliotecas estándar de C (es decir, no debe usar bibliotecas como `conio.h` o llamadas directas a `system`).
9. El sistema debe ser robusto, es decir, sobreponerse a los errores esperables. Se penalizarán los errores no manejados.

IV. Entrega

1. Debes subir tu trabajo al aula virtual de tu curso de cátedra dentro de la plataforma `https://unab.blackboard.com/`, en una casilla que se habilitará especialmente para esto.
2. No se aceptarán trabajos atrasados.
3. El trabajo debe ser subido por solo uno (1) de los integrantes del grupo.
4. Debes subir **solo el código fuente**.

5. El nombre del archivo con el código fuente debe seguir el formato `rutCompleto1_rutCompleto2`, con la extensión adecuada, donde `rutCompletoX` corresponde al RUT de cada integrante del grupo, sin puntos ni guión, pero incluyendo el dígito verificador.

6. Si el programa no compila o no se puede ejecutar, tendrá nota mínima (1,0). Si funciona, se evaluará ejecución y código fuente.

7. Ante el escenario de existir sospecha de copia (con otros compañeros, o desde internet), será interrogado acerca de su trabajo, para aclarar dudas de su entendimiento y autoría. Si se confirma la sospecha, el trabajo será evaluado con nota 1,0.

8. Las consultas las debe realizar directamente a los profesores de taller, presencialmente en clases o a los correos y en los horarios que ellos establezcan.