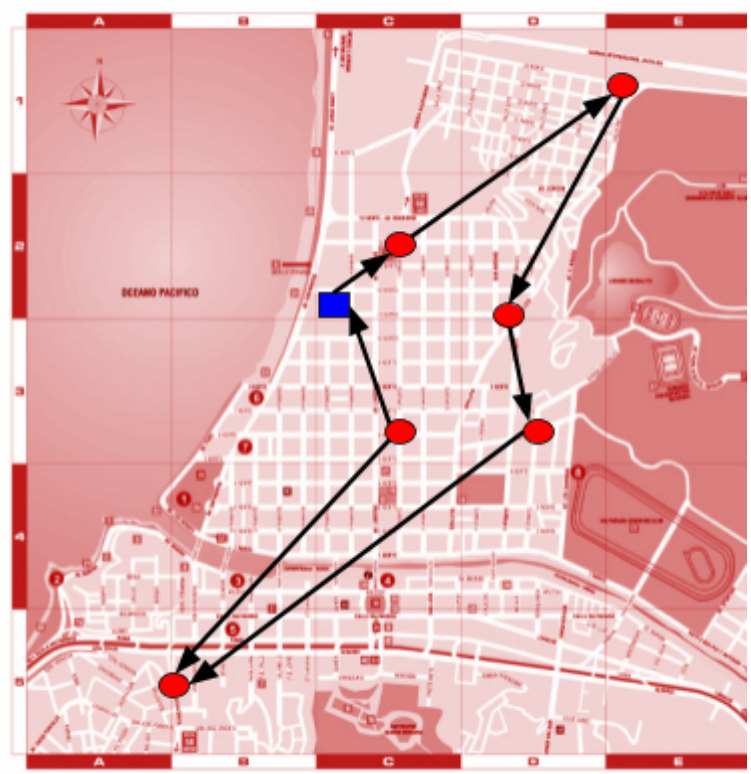
Tarea 3

Objetivos

- Uso de grafos en el lenguaje de programación C.
- Diseñar y proponer una solución eficiente para el problema planteado.

Actividad

Durante la pandemia, Javiera creó un emprendimiento de venta de productos. Importa productos desde China y los vende por internet. Además, realiza entregas diarias de productos, para ello, toma un mapa y crea una ruta usando las direcciones de los clientes, sin embargo la ruta que genera no es óptima y además le toma mucho tiempo hacer una ruta diaria.



Es por ello, que le pide a ud. que desarrolle una aplicación con las siguientes opciones:

1. **Importar archivo de coordenadas (char* nombre_archivo, int n)**: La aplicación importa las **n primeras líneas** de un archivo que contiene las coordenadas asociadas a cada entrega. El archivo tiene el siguiente formato (tsp.txt):

```
6734 1453
2233   10
5530 1424
 401  841
3082 1644
...
```

Cada línea corresponde a las coordenadas de una entrega. Como identificador de la entrega se usa el número de línea (por ejemplo, las coordenadas de la entrega **2** son: 2233,10)

2. **Distancia entre entregas (int entrega1, int entrega2)**: La aplicación muestra la distancia (en línea recta) entre las 2 entregas.
3. **Mostrar 3 entregas más cercanas a las coordenadas ingresadas (int x, int y)**: La aplicación muestra el id y la distancia a las 3 entregas más cercanas (de la más cercana a la más lejana).
4. **Crear ruta (int x, int y)**: El usuario ingresa sus coordenadas (x,y) y la aplicación muestra el resto de las entregas ordenadas de menor a mayor distancia. El usuario ingresa el id de una entrega. Luego la aplicación muestra las distancias a las entregas faltantes (de menor a mayor distancia) y la distancia total recorrida hasta el momento. El usuario escoge una segunda entrega y así sucesivamente hasta que todas las entregas hayan sido realizadas. Al terminar la ruta, la aplicación muestra la secuencia de entregas y la distancia total recorrida (agregando la distancia entre la última entrega y las coordenadas iniciales). Finalmente, la aplicación pide al usuario que le ponga un nombre a la ruta generada.
5. **Generar ruta aleatoria (int x, int y)**: Se genera una ruta aleatoria, se muestra la secuencia generada y la distancia recorrida. La aplicación le pide al usuario que le ponga un nombre a la ruta generada.
6. **Mejorar ruta (char* nombre)**: La aplicación muestra la información de la ruta indicada por el usuario. El usuario puede decidir si realizar un intercambio manual o automático (aleatorio) entre 2 entregas y ver si el resultado mejora. Por ejemplo:
   Ruta: 1-2-3-4-5-6, distancia recorrida: 120
   Intercambiar 2 con 6.
   Ruta nueva: 1-6-3-4-5-2, distancia recorrida: 100
   Si la nueva ruta reduce la distancia, la ruta es modificada. Si empeora el resultado, se vuelve al estado/ruta original.
7. **Mostrar rutas ()**: La aplicación muestra todas las rutas guardadas (de la mejor a la peor).
8. **Mejor ruta (int x, int y)**: El usuario ingresa sus coordenadas (x,y) y la aplicación genera la ruta que realiza todas las entregas y **minimiza la distancia recorrida**. La ruta es guardada con el nombre de "ruta óptima".

Sugerencias

- Para las **opciones 4, 5, ...,** se sugiere que utilice un grafo implícito. El **estado** representaría una ruta completa/incompleta, es decir: la secuencia de entregas realizadas hasta un momento dado, las entregas que faltan por realizar y la distancia recorrida hasta el momento. Una **acción** correspondería a agregar una nueva entrega al estado. Cada vez que el usuario ingresa una nueva entrega, se estaría realizando una **transición**. Notar que un **estado final** sería equivalente a una ruta completa.
- Para la **opción 8**, se sugiere que implemente un algoritmo de búsqueda usando el mismo grafo implícito de la opción 4.
- Para la **opción 8**, se sugiere que pruebe con una pocas entregas (e.g., 5) y vaya aumentando. La complejidad de los algoritmos de búsqueda es generalmente exponencial por lo que para más de 10 entregas probablemente se tarde demasiado en encontrar la solución óptima.

Archivo README.md

Además del código, en el repositorio debe adjuntar un archivo README.md que describa:

- Opciones que **no funcionen correctamente** en el programa indicando las posibles causas con claridad.
- Aspectos positivos y a mejorar por cada uno de los integrantes (al menos 2 de cada uno).
- Puntos de premio o castigo para cada integrante del equipo (**deben sumar 0**). Estos puntos equivalen a décimas que se agregan o descuentan de la nota obtenida.

Consideraciones

1. La tarea se puede realizar en equipo.
2. Recuerde además que las revisiones de avance en clases valen un **20% de la nota**.
3. **Copia será calificada con nota 1.0.**
4. **Tareas que no compilen serán calificadas con nota 1.0.** (Sugerencia: entregue poco y que funcione bien a mucho y que no funcione.)
5. Utilice librerías estándar de C (ver aquí).
6. La fecha límite de entrega es el **30 de mayo, hasta las 23:59 horas.**
7. **Entregas atrasadas serán penalizadas con 1 pto menos por cada hora de atraso.**
1. La tarea debe ser **subida a un repositorio en github**. Debe dejar el link en el aula virtual.