UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2014

19 de octubre de 2014

Ejercitación sobre Entrada/Salida

1. Ejercicios a resolver

Ejercicio 1.

1. Implementar el método `imprimirCiudad()` de la clase Ciudad, que dado un output stream y una ciudad, la imprima por el stream pasado por parámetro utilizando el formato adecuado.
2. Implementar el método `imprimirMapa()` de la clase Mapa, que dado un output stream y una ciudad, lo imprima por el stream pasado por parámetro utilizando el formato adecuado.

Ejercicio 2.

1. Implementar el método `leerCiudad()` de la clase Ciudad, que lea una ciudad de un input stream y la guarde en la ciudad pasada por parámetro.
2. Implementar el método `leerMapa()` de la clase Mapa, que lea un mapa desde un input stream y lo guarde en un mapa pasado por parámetro.

Ejercicio 3. Hacer un menú interactivo que pueda responder a las siguientes operaciones:

- leer mapa de un archivo (se utilizará el comando **leerMapa** seguido del *path* del archivo desde donde leerlo)
- imprimir mapa a un archivo (se utilizará el comando **cargarMapa** seguido del *path* del archivo adonde imprimir)
- imprimir mapa por pantalla (se utilizará el comando **imprimirMapa**)
- agregar una ruta entre dos ciudades (se utilizará el comando **agregarRuta** seguido de dos ciudades con el formato correspondiente)
- salir del programa (se utilizará el comando **salir**)

2. Formato pedido de Mapa y Ciudad

2.1. Formato Ciudad

En una sola línea, se colocará primero el nombre de la ciudad (sin espacios) y seguidamente la cantidad de habitantes de la misma, separados por punto y coma y expresados como un par.

Ejemplos:

```
(BuenosAires;4000000)
(Rosario;1234567)
```

2.2. Formato Mapa

En la primera línea se deben colocar en orden la cantidad de ciudades y el número total de rutas que hay en el mapa.

En la segunda línea, como se muestra en el ejemplo, se deberá dar una lista de todas las ciudades que se encuentran en el mapa. No podrá haber dos ciudades con el mismo nombre. Cada ciudad respetará el formato descrito en la sección anterior. Notar que la cantidad de ciudades debe coincidir con el número de ciudades de la primera línea.

En la tercera línea se deben enumerar todas las vecindades del mapa en una lista, mostrando solamente los nombres de las ciudades vecinas (separados por coma, y entre llaves).

Notar que se deberá imprimir tanto {u,v} como {v,u}, y que no importa el orden en el que sean impresas las parejas.

```
3 4
[(Catamarca;100000),(Salta;129312),(Neuquen;129387)]
[{Catamarca,Salta},{Salta,Catamarca},{Catamarca,Neuquen},{Neuquen,Catamarca}]
```

3. Algunas funciones útiles

3.1. Convertir un string a int

```
cantHabsInt = atoi(cantHabsString.c_str());
```

si `cantHabsString` es el string que queremos convertir a entero.
Nota: también pueden implementarlo ustedes mismos como una función auxiliar, es un muy buen ejercicio.

3.2. Leer de un stream hasta encontrar cierto caracter

Por ejemplo, si nuestra entrada es `Alice-Bob` y queremos leer un string que solamente contenga `Alice`, podemos hacer lo siguiente:

```
getline(inputStream,stringDondeGuardo,'-');
```

Esto indica que se debe leer de un stream (y guardarlo en el string dado) hasta que se encuentre un char guion. Allí se detendrá de leer, y descartará el guion leído. Lo pueden utilizar con otros caracteres, como por ejemplo la coma, o el punto y coma.

Aclaración: pueden asumir que todos los archivos que se utilicen no contendrán espacios, por lo que pueden usar *cin* >> ... sin problemas.