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

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2011

4 de noviembre de 2011

TPI - Esemeseando

1. Introducción

En el TPE, hemos especificado el comportamiento de una serie de problemas relacionados con usuarios de telcos que participan en trivias. Después hicimos una primera implementación del trabajo en programación funcional en el TPF. Nuestro próximo paso será obtener una implementación en un lenguaje imperativo, para lo cual la especificación original ha sido *especialmente* adaptada.

El objetivo de este trabajo es programar, usando los comandos de C++ vistos en clase, tipos y operaciones análogos a los de los TPs anteriores, pero con una especificación que tiene en cuenta el uso de clases y características propias del paradigma imperativo.

Deberán utilizar, además, el tipo abstracto `Lista`, provisto por la cátedra. Publicaremos la interfaz del tipo y su implementación. Para hacer uso de él, sólo deben incluir el archivo .h correspondiente. No hay un .cpp. NO deben mirar la implementación. Sólo deben manejarlo en base a su especificación.

También deberan utilizar fuertemente el tipo `pair` provisto por la `std`. Pueden encontrar una buena referencia para su uso en `http://www.cplusplus.com/reference/std/utility/pair/`

Por ï¿½ltimo, para implementar las funciones pedidas en auxiliares.h (y sï¿½lo para esto) deberï¿½n usar el tipo `string`, tambiï¿½n provisto por la `std`. La referencia del mismo se encuentra en `http://www.cplusplus.com/reference/string/string/`. No se permite usar los mï¿½todos de string en el resto del TP.

En la página de la materia estarán disponibles los archivos mencionados arriba, los *headers* de las clases que deberán implementar y la especificación de cada uno de sus métodos. **Importante: Utilizar la especificación diseñada para este TP, no la solución del TPE!**

Pueden agregar los métodos auxiliares que necesiten. Estos deben estar *en todos los casos* en la parte privada de la clase. No se permite modificar la parte pública de las clases ni agregar atributos adicionales, ya sean públicos o privados. No se permiten archivos "extra" con funciones auxiliares, ni tampoco modificar el archivo auxiliares.h.

Para que la fiesta sea completa, tendrán disponible un sistema básico de menúes que les permitirá utilizar sus clases en forma interactiva. Este menú estará disponible en la página de la materia (interfaz.cpp e interfaz.h) junto al resto de los archivos. Si desean mejorarlo, no hay inconveniente que lo hagan pero no será evaluado. El archivo esemeseando.cpp, que también proveemos, contiene la función main que llama a dicho menú.

2. Demostraciones

Deben implementar la función `infoGatewaysTE` dentro del archivo esemeseando.cpp y demostrar su terminación y correctitud. De utilizar funciones auxiliares para su implementación, si las mismas **NO** se encuentran como problemas en la solución de la cátedra, deberán especificarlas y demostrar terminación y correctitud de ellas también.

`problema infoGatewaysTE` (te: TelCO) = `result` : [(Gateway, ($\mathbb{Z}$, $\mathbb{Z}$))] {
  `asegura infoCorrecta` : $mismos(result, [(g, (|trivias(g)|, cuantosJueganAlgunaTrivia(te, g)))|g \leftarrow gateways(te)])$;
}

Observación: Tomar la especificación de las funciones auxiliares del documento provisto por la materia.

3. Entrada/Salida

Todas las clases del proyecto tienen tres métodos relacionados con entrada salida:

mostrar : que se encarga de mostrar todo el contenido de la instancia de la clase en el flujo de salida indicado. El formato es a gusto del consumidor, pero esperamos que sea algo más o menos informativo y legible.

guardar : que se encarga de escribir la información de cada instancia en un formato predeterminado que debe ser decodificable por el método que se detalla a continuación (`cargar`).

cargar : que se encarga de leer e interpretar información generada por el método anterior, modificando el valor del parámetro implícito para que coincida con aquel que generó la información.

En definitiva, `guardar` se usará para "grabar" en un archivo de texto el contenido de una instancia mientras que "cargar" se usará para "recuperar" dicha información. En todos los casos, sus interfaces serán:

- `void mostrar(std::ostream& ) const;`
- `void guardar(std::ostream& ) const;`
- `void cargar(std::istream& );`

El detalle del formato que se debe usar para "guardar" y "leer" en cada clase se indica a continuación. Tener en cuenta que los números se deben guardar como texto. Es decir, si escriben a un archivo y después lo miran con un editor de texto, donde escribieron un número 65 deben ver escrito 65 y no una letra A. Cada vez que aparezca un string, éste deberá ir guardado entre `|`. Pueden suponer que los nombres no contendrán el caracter `|`.

SMS: Se debe guardar el `ENCABEZADO_ARCHIVO`, el numero y entre barras (barrita vertical `"|"`) el texto. Por ejemplo:

```
S 2345 |Salgamos a tomar algo|
```

Pregunta: Se debe guardar el `ENCABEZADO_ARCHIVO` y entre barras la pregunta y la respuesta, y el puntaje sin delimitar. Por ejemplo:

```
P |Quien gano el mundial 78| |Argentina| 2
```

Trivia: Se debe guardar el `ENCABEZADO_ARCHIVO`, entre barras las keywords, luego las preguntas y por ï¿½ltimo los elementos con formato (usuario, puntos, prox_pregunta), donde prox_pregunta contendrï¿½ el nï¿½mero total de preguntas en caso de que el jugador sea ganador. Por ejemplo, :

```
V [|futbol|,|gol|] [P |el 10| |maradona| 10, P |ganador mundial 78| |Argentina| 5]
  [(9874,5,2),(3589,10,1)]
```

Donde el jugador 1 es ganador y el jugador 2 sï¿½lo respondio una pregunta. **Aclaración 1:** en este tipo, y en todos los que involucran listas, no está permitido usar mostrar de Lista para guardar. Sí está permitido usarlo para mostrar. Las listas deben ir encerradas entre corchetes y con sus elementos separados por comas.

Gateway: Se debe guardar el `ENCABEZADO_ARCHIVO`, el nï¿½mero, los comandos y las trivias.
es decir: G nro [comandos][trivias]

A continuación un ejemplo de gateway:

```
G 1234
[|Baja|,|Ranking|,|jugar|]
[V [|futbol|,|gol|] [P |el 10| |maradona| 10, P |ganador mundial 78| |Argentina| 5]
   [(9874,5,2),(3589,10,1)]
 V [|signos|] [P |15 agosto| |Leo| 3, P |26 septiembre| |Libra| 8] [(1235,3,1),(9874,0,1)]
]
```

Importante 1: Los saltos de línea han sido dejado por cuestiones de visualización. En el archivo real no deben incorporarse!

Importante 2: Observar que los nombres de productos y personas siempre van entre barras.

Telco: Se debe guardar el `ENCABEZADO_ARCHIVO`, los usuarios, sus gateways y por ultimo los mensajes con sus respectivas respuestas.
es decir: T [usuarios] [gateways] [mensajes]

A continuación un ejemplo de TelCO:

```
T [1235,3589,9874,2313]
[G 1234
 [|Baja|,|Ranking|,|jugar|]
 [V [|futbol|,|gol|] [P |el 10| |maradona| 10, P |ganador mundial 78| |Argentina| 5]
    [(9874,5,2),(3589,10,1)],
  V [|signos|] [P |15 agosto| |Leo| 3, P |26 septiembre| |Libra| 8] [(1235,3,1),(9874,0,1)]
 ]
]  [(S 1234 |gol maradona|, S 9874 |Bien! Tenés 10 pts. ganador mundial 78|),
    (S 1234 |signos Aries|, S 3589 |Mal! Tenés 0 pts. 26 septiembre|),...
   ]
```