Algoritmos y Programación II

Trabajo práctico N° 4:

"Botánica binaria"

Idea y guión: Nicolás Marianetti

Objetivo:

En este trabajo se busca que los alumnos de Algoritmos y Programación II sean capaces de resolver un problema de la vida real aplicando sus conocimientos en:

- Herencia
- Polimorfismo
- Árboles

Problema:

La empresa de telefonía "Andy & Co" quiere mejorar la eficiencia de su sistema de:

- Marketing
- Alta / Baja de clientes
- Búsqueda de clientes

Para esto, le piden a un alumno de Algoritmos y Programación II que cree un programa orientado a objetos donde puedan:

- Saber cuánto será el precio de un producto que van a ofrecer.
- Agregar un nuevo cliente.
- Dar de baja un cliente.
- Listar clientes.
- Buscar, mediante un número de teléfono, un cliente.

Aspectos técnicos:

- Los clientes de la empresa se encuentran en un archivo llamado **clientes.txt**, en este archivo también se encuentra el número de teléfono vinculado a un cliente.
- Los clientes de la empresa pueden ser **individuos** o **familias**. Las familias están compuestas por dos o más personas, los individuos, solo tienen una persona.
- Los números de teléfono tienen un formato: **12345678**
- Dependiendo del tipo de cliente, se aplica un descuento **fijo**. Este descuento es aplicado a un **precio base** que ingresa la persona que utilice el programa. El descuento varía según el cliente, siendo:

- 10%, si es un individuo
- 35%, si es una familia

- Si se desea agregar un nuevo cliente, se le pide nombre (si es un individuo) o nombres (si es una familia), luego se le asigna un número de teléfono automáticamente, éste es siempre el mismo y es de la forma:

  00X

  (Siendo X el número de padrón de cualquiera de los integrantes del grupo)

  Ej: 00**101586** o 000**99846**

  Si este número se encuentra ocupado, se le intenta otorgar el siguiente.

  Ej: Intenté con 00101158**6** y estaba ocupado, intento con 00101158**7**; este también estaba ocupado, intento con 00101158**8** y así…
- Para dar de baja a un cliente, se utiliza un número de teléfono (ya sabido por el operador del programa). Si este número de teléfono no se encuentra vinculado a ningún cliente, el sistema no debe hacer ninguna modificación e informará al operador del programa que el número de teléfono no se encuentra vinculado a un cliente.
- Al listar clientes, se espera que se muestre por pantalla algo de la forma:

  ```
  12345678 Michael Jackson
  13456789 Arya Stark Marilyn Monroe
  …
  ```

  Nota: los números de teléfono deben aparecer en forma ascendente

- Al buscar un cliente, se debe mostrar por pantalla el / los nombre/s y apellido/s de la/s persona/s vinculada/s a un número de teléfono. En caso de que no haya un cliente vinculado a un número de teléfono, se deberá informar al operador del programa que el número de teléfono no se encuentra vinculado a un cliente.

Formato del archivo:

El archivo será un CSV con el siguiente formato:

telefono,nombre_1,nombre_2,…,nombre_n

Por ejemplo:

```
12345678,Juan Perez
55555555,Rosa Romero,Pedro Perez,Juan Perez Romero,María Perez Romero
etc
```

Tener en cuenta:

- Cada línea del archivo tiene un número de teléfono y por lo menos un nombre. Si hay más de un nombre, es una familia.

  Para detectar si hay un solo nombre o más, recomendamos leer el archivo con la función *getline* que lee toda una línea del archivo y luego ir buscando las comas y separando el string usando una combinación de los métodos *find* y *substr.* Al final hay un ejemplo de su uso.
- La estructura donde se irán cargando debe ser un ABB (árbol binario de búsqueda), es decir, los números de teléfono mayores a la raíz irán en el subárbol derecho y los menores en el izquierdo.
- Los nodos tendrán:
  - Los dos punteros a los subárboles.
  - El número de teléfono.
  - Un puntero a los datos, ya sea un individuo o una familia.

Consideraciones:

- Buenas prácticas de programación: código, modularización, comentarios, claridad.
- Uso de herencia, polimorfismo y árboles.
- Memoria dinámica.
- Pre y post condiciones
- Interfaz de usuario

- El modelo de solución debe respetar el principio Open/Closed (pensarlo en funcióon de que el día de mañana la empresa quiera agregar un nuevo tipo de cliente, ver en https://devexperto.com/principio-open-closed/).

Normas de entrega

El ejercicio debe ser desarrollado en los equipos que están formados.

Deben entregar:

- Documentación
  - Diagrama de clases UML
  - Diagrama de relación de clases.
  - Descripción de cada TDA, indicando las pre y poscondiciones de cada una de las operaciones.
- Código fuente
  - En los archivos .h también van las pre y poscondiciones.

La fecha de entrega vence el viernes 21/6 a las 23.55hs

Ejemplos de uso métodos find y substr

```
string s = "hola Maria, que tal, todo bien?";
unsigned pos = s.find(',');         // devuelve la posición de la primera coma
string s2 = s.substr(0, pos);       // s2 = Hola Maria (string desde 0 hasta pos)
string s3 = s.substr(pos + 1);      // corta desde el espacio hasta el final
cout << s3 << endl;                 // Imprime " ,que tal, todo bien?"
pos = s.find(',', 20);              // no hay comas después de la posición 20

if (pos != string::npos)
    cout << "La siguiente coma esta en la posicion: " << pos << endl;
else
    cout << "No se encontro ninguna mas" << endl;
```

Para más información consultar en reference cplusplus:

- Find: http://www.cplusplus.com/reference/string/string/find/
- Substr: http://www.cplusplus.com/reference/string/string/substr/