Práctica Final: Quién es quien

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

DECSAI

Estructuras de Datos

Grado en Ingeniería Informática
Grupo C

Índice de contenido

1.Introducción....3
2.Objetivo....3
3.Ejercicios....3
3.1. Tareas a realizar....3
3.2.Codificación de los rasgos....4
3.3.Programas....5
3.3.1.Test....5
3.3.2.WhoisWho....6
3.3.3.Suficiencia del conjunto de preguntas....9
3.3.4.Conjunto minimal de preguntas....9
3.4.Ficheros....10
3.4.1.Fichero de Configuración....10
3.4.2.Fichero tree....11
4.Módulos a desarrollar....12
5.Práctica a entregar....13
6.Referencias....14

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

- Resolver un problema eligiendo la mejor estructura de datos para las operaciones que se solicitan

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 2: Abstracción de datos. Plantillas.
3. Haber estudiado el Tema 3: T.D.A. Lineales.
4. Haber estudiado el Tema 4: STL e Iteradores.
5. Haber estudiado estructuras de datos jerárquicas: Árboles

2. Objetivo.

El objetivo de esta práctica es llevar a cabo el análisis, diseño e implementación de un proyecto. Con tal fin el alumno abordará un problema donde se requiere estructuras de datos que permiten almacenar grandes volúmenes de datos y poder acceder a ellos de la forma más eficiente.

3. Ejercicios

El alumno deberá implementar varios programas. Uno de ellos simulará al famoso juego "Quien es quien". Este juego, con dos concursantes, consiste en descubrir el personaje oculto de su adversario. Cada concursante parte del mismo conjunto de personajes (ver Figura 1) y de un personaje oculto (elegido entre el conjunto de personajes). La dinámica del juego consiste en que un concursante efectúa una pregunta acerca del personaje oculto del otro y este le responde afirmativamente o negativamente. Según la respuesta dada el concursante con el turno eliminará de su conjunto de personajes aquellos que no respondan a la pregunta hecha. A continuación el turno pasa al otro concursante. Aquel concursante que descubre antes el personaje oculto de su contrario gana.

3.1. Tareas a realizar

El alumno debe llevar a cabo las siguientes tareas:

1. Con objeto de que alumno practique con el T.D.A ArbolGeneral, se pide que implemente este tipo de dato. Este tipo de dato será necesario ya que los rasgos del conjunto de personajes vienen codificados en un árbol general (como se muestra en la Figura 2). En la sección 3.2 se detallará más la codificación de los rasgos de los personajes.
2. Definir el resto de T.DA. que vea necesario para la solución de los problemas propuestos.
3. Probar los módulos con programas test.
4. Construir los programas que a continuación se detallan.

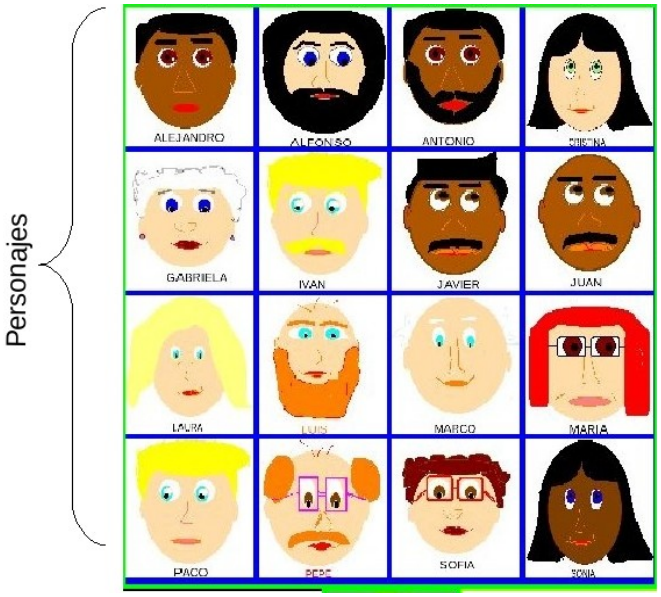


Figura 1: Imagen de los personajes del juego

Se puede usar la STL en todos los módulos excepto en la implementación del TDA ArbolGeneral.

3.2. Codificación de los rasgos

A cada personaje le daremos un código que va desde (*0...n-1*) siendo n el número de personajes. Los rasgos de los personajes se codificarán en un árbol general. El árbol general en cada nivel se plantea la misma pregunta (ver Figura 2), p.e ¿Tiene bigote?. Ante esa pregunta las posibilidades son dos, sí o no. Cada nodo interior (que no es hoja) se corresponde con una pregunta que se pueda hacer (dadas en el fichero de configuración ver sección 3.4.1). Los nodos hojas contiene valores en el rango (*-1 0 ....n*). El valor -1 indica que ningún personaje responde a la secuencia de preguntas que se han hecho desde la raíz hasta el nodo padre de la hoja actual (con valor -1). Cualquier otro valor (*0...n-1)* se corresponde con la clave del personaje. Tened en cuenta que una secuencia de preguntas y sus correspondientes respuestas dará lugar a ninguno, uno o más de un personaje con esas características.

Así por ejemplo el siguiente sub-camino que se puede ver en la Figura 2:

....¿Es Oscura su color de piel? Si- ¿Es Calvo? Si- ¿Tiene bigote? Si- ¿Tiene barba? No

Da lugar a la descripción del personaje "Juan". Este camino podría codificarse como una secuencia de ceros y unos. Cero para ir por la respuesta No y 1 para ir por la respuesta Si. Por ejemplo el anterior subcamino se puede codificar como :

....1110

Si un nodo hoja, que se corresponde con unos rasgos determinados, tiene hermanos significa que distintos personajes responden a esos rasgos (podríamos decir que una secuencia de rasgos se corresponden con muchos personajes). Y por lo tanto el conjunto de preguntas realizadas ***no es suficiente*** para discernir entre los personajes.

Por otro lado si con menos preguntas somos capaces de describir a todos los personajes

entonces diremos que el conjunto de preguntas ***no es minimal.***

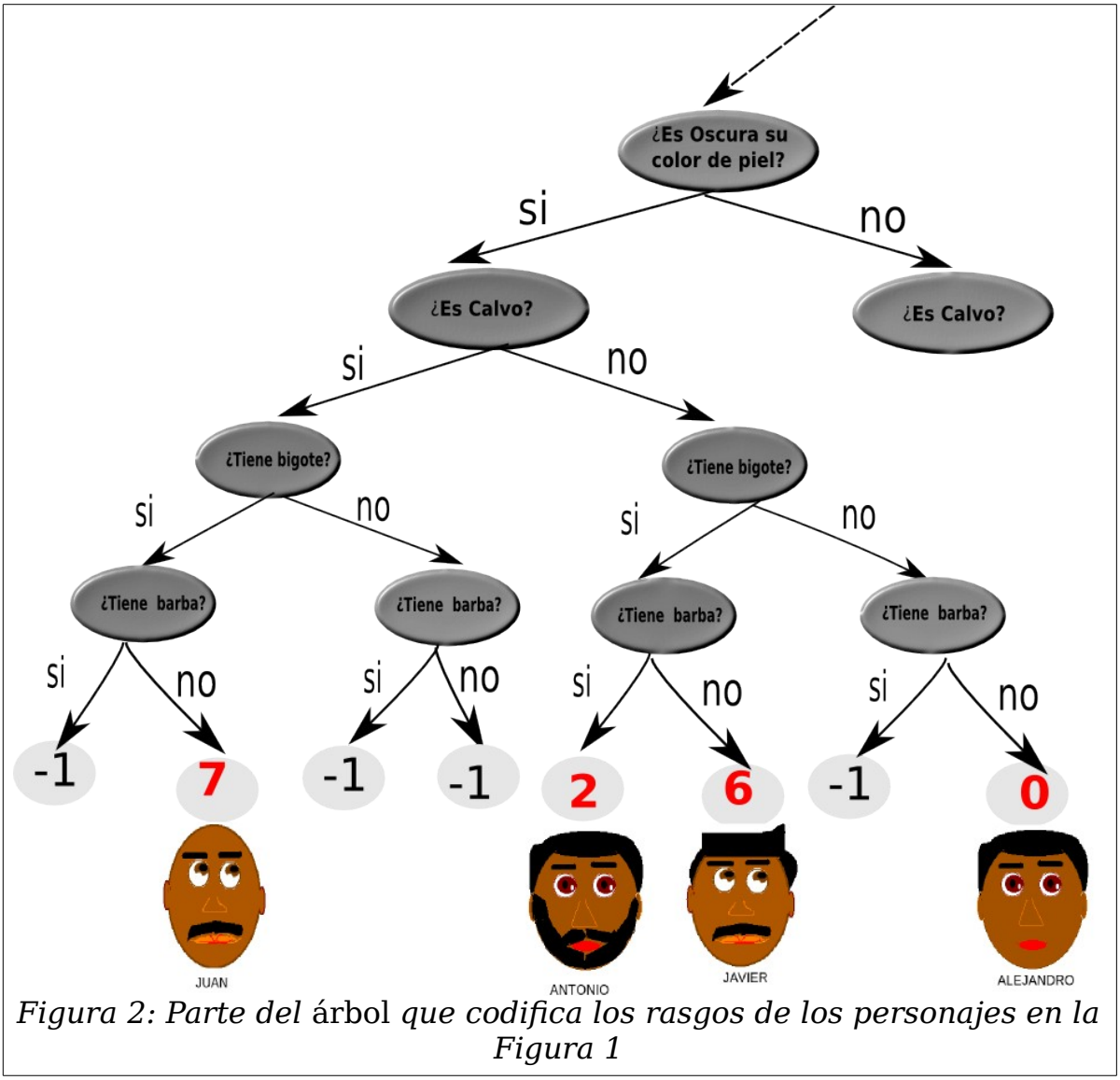


Figura 2: Parte del árbol *que codifica los rasgos de los personajes en la Figura 1*

A continuación detallaremos cada uno de los programas.

3.3. Programas

3.3.1. Test

En este código, lee del fichero de configuración que contiene las preguntas posibles, personajes y nombre del fichero con el árbol de rasgos (ver sección 3.4.1). El programa imprimirá por cada personaje las respuestas a cada una de las preguntas y además dará el código como una secuencia de ceros y unos a esas preguntas. El alumno creará por lo tanto el programa test que se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  bin/test datos/caras1.txt
```

A modo de ejemplo en el terminal aparecerá lo siguiente:

```
Personaje: Alfonso
Preguntas y contestaciones son:
¿Es Corto su pelo?Si
¿Es Largo su pelo?No
¿Es Hombre?Si
¿Es Mujer?No
¿Tiene gafas?No
¿Tiene la piel Oscura?No
¿Tiene la piel Clara?Si
¿Tiene el pelo Marron?No
¿Tiene el pelo Pelirrojo?No
¿Tiene el pelo Blanco?No
¿Tiene el pelo Rubio?No
¿Tiene el pelo Negro?Si
¿Es Oscuro su color de ojos?No
¿Es Claro su color de ojos?Si
¿Es calvo?No
¿Tiene bigote?Si
¿Tiene barba?Si
Codigo asociado 11010100001000101
…
```

Esta misma salida se repetirá para cada uno de los personajes.

El fichero test.cpp será suministrado en el material asociado a la práctica. Este código no deberá ser cambiado así que deberá funcionar con los TDA desarrollados por el alumno. En particular para este programa el alumno tendrá que implementar el TDA ArbolGeneral. Para facilitar la tarea al alumno se le ha dado implementado el operador de incremento para el iterador en preorden.

3.3.2. WhoisWho

Este programa cuando es ejecutado implementa el juego quién es quién. Para poder ejecutar el programa en la línea de ordenes escribiremos:

```
prompt%  bin/whoiswho datos/caras1.txt h
```

Los parámetros de entrada son los siguientes:

1. El nombre del fichero con la configuración del juego
2. Un carácter que puede ser **h** o **m. h** si el juego será entre la máquina y un humano. O **m** si el juego sera entre la máquina consigo misma (usando diferentes estrategias).

Tras la ejecución en pantalla aparecerá lo siguiente:

```
Tu personaje es :Ivan
Elige una de las siguiente preguntas:
1.-¿Tiene barba?
2.-¿Tiene bigote?
3.-¿Es calvo?
4.-¿Es Claro su color de ojos?
5.-¿Es Oscuro su color de ojos?
6.-¿Tiene el pelo Negro?
7.-¿Tiene el pelo Rubio?
8.-¿Tiene el pelo Blanco?
9.-¿Tiene el pelo Pelirrojo?
10.-¿Tiene el pelo Marron?
11.-¿Tiene la piel Clara?
12.-¿Tiene la piel Oscura?
13.-¿Tiene gafas?
14.-¿Es Mujer?
15.-¿Es Hombre?
16.-¿Es Largo su pelo?
17.-¿Es Corto su pelo?
18.-Digo ya el personaje
¿Cual eliges?15
humano>¿Es Hombre?
maquina>No
************************
maquina>¿Es Claro su color de ojos?
humano>Si
************************
Elige una de las siguiente preguntas:
1.-¿Tiene barba?
2.-¿Tiene bigote?
3.-¿Es calvo?
4.-¿Es Claro su color de ojos?
5.-¿Es Oscuro su color de ojos?
6.-¿Tiene el pelo Negro?
7.-¿Tiene el pelo Rubio?
8.-¿Tiene el pelo Blanco?
9.-¿Tiene el pelo Pelirrojo?
10.-¿Tiene el pelo Marron?
11.-¿Tiene la piel Clara?
12.-¿Tiene la piel Oscura?
13.-¿Tiene gafas?
14.-¿Es Mujer?
15.-¿Es Hombre?
16.-¿Es Largo su pelo?
17.-¿Es Corto su pelo?
18.-Digo ya el personaje
¿Cual eliges?18
Dime el nombre del personaje:Gabriela
maquina>Si
Has ganado
```

En primer lugar el programa le muestra al jugador humano cual es su personaje oculto y además le lista todas las preguntas que leyó del fichero de configuración. En el recuadro anterior las preguntas son desde la 1 a 17. La 18 se añade para que el jugador humano pueda indicar el nombre del personaje oculto de la máquina cuando piense que lo ha adivinado. Para ayudar a la detección del personaje oculto de la máquina el usuario tiene a su disposición en el directorio datos/caras1/ el fichero caras1.svg. En este fichero se muestra todos los personajes y si se pincha sobre alguno de ellos se activa o desactiva. Para ver este fichero podemos ejecutar

```
prompt%  firefox datos/caras1/caras1.svg
```

Módulo: la inteligencia de la máquina

En este apartado queremos hacer mejor jugador a la máquina. El jugador máquina juega mejor si es más rápido en adivinar el personaje oculto de su contrario. Esta rapidez dependerá de la táctica (heurística) escogida para realizar la pregunta al contrario. Por

ejemplo la táctica más fácil de programa es escoger una de las preguntas, que aún no he realizado, de forma aleatoria. Pero existen mejores heurísticas que se propone al alumno que programe de forma voluntaria:

- Escoger la pregunta de mayor entropía. La entropía se maximiza cuando el conjunto original se divide en dos subconjuntos con igual numero de personajes. Uno de los conjuntos contestando Sí a la pregunta y otro conjunto contestando No. De forma matemática la entropía se formula como:

$$H(x) = -\sum p(x_i)\log_2(p(x_i)) \quad \text{donde}$$

  - x es la pregunta
  - x_i son las posibles respuesta. En nuestro caso son dos : "Sí"o "No".
  - $p(x_i)$ es la probabilidad de que se dé la respuesta x_i a la pregunta x con un subconjunto del personajes (los no descartados del conjunto de partida).

Para aclarar mejor los conceptos veamos un ejemplo con mayor detalle. Suponiendo el conjunto de personajes de la figura 1 entre las siguiente dos preguntas:

1. ¿Es Mujer su Sexo?
2. ¿Es Claro su Color de Ojos?

Pasamos a calcular la entropía de cada una de las preguntas

$$H(\text{¿ Es Mujer Su Sexo ?}) = -(p(No)*\log_2(p(No)) + p(Si)*\log_2(p(Si)))$$

donde las probabilidades son:

$$p(No) = \frac{10}{16} \quad \text{y} \quad p(Si) = \frac{6}{16}$$

Si hacemos el cálculo obtenemos que

$$H(\text{¿ Es Mujer Su Sexo ?}) = 0.954434$$

y

$$H(\text{¿ Es Claro su Color de Ojos ?}) = -(p(No)*\log_2(p(No)) + p(Si)*\log_2(p(Si)))$$

donde las probabilidades son:

$$p(Si) = \frac{9}{16} \quad \text{y} \quad p(No) = \frac{7}{16}$$

Haciendo el cálculo de la entropía obtenemos que

$$H(\text{¿ Es Claro su Color de Ojos ?}) = 0.9886$$

Por lo tanto la máxima entropía se obtiene con la pregunta *¿Es Claro su Color de Ojos?*. Intuitivamente, alta entropía quiere decir que nos acercamos más a la equiprobabilidad de respuestas, o dicho de otra forma: más o menos tenemos el mismo número de respuestas en ambos sentidos. Esto es bueno porque garantiza que, en el peor de los casos, descartamos el mayor número posible de personajes.

Si elegimos preguntas con baja entropía, puede ser que tengamos suerte y descartemos muchos personajes con una pregunta pero si somos poco afortunados descartaremos muy pocos. Si encadenamos varias preguntas "con mala suerte" el resultado será muy malo. Esta es una mala estrategia, ya que lo habitual es pensar siempre en la situación más desfavorable para decidir qué hay que hacer.

Hay que tener en cuenta que no deberíamos volver a realizar preguntas ya hechas. Con esto último se quiere hacer notar que, por ejemplo, si el jugador preguntó *¿Es Claro su Color de Ojos?* y la contestación fue afirmativa, en el futuro no se volverá a hacer la misma pregunta.

3.3.3. Suficiencia del conjunto de preguntas

Dado un conjunto de preguntas y un conjunto de personajes con unos rasgos que vienen codificados en un árbol general, se pide desarrollar un programa que indique al usuario si con ese conjunto de personajes y ese conjunto de preguntas se puede determinar de forma unívoca a cada uno de los personajes. Esta tarea es fácil ya que teniendo la codificación de los rasgos de los personajes como una cadena de ceros y unos, simplemente tenemos que ver si dos códigos son iguales. El alumno creará por lo tanto el programa test que se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  suficiente  datos/conjunto_no_suficiente.txt
```

Para este ejemplo la salida del programa sera :

"Las respuestas son las mismas para los personajes: Sonia y Cristina"

Y por lo tanto no es suficiente.

3.3.4. Conjunto minimal de preguntas

Un conjunto de preguntas es minimal si al eliminar de este conjunto alguna pregunta ya el conjunto de preguntas no es suficiente (es decir no podemos discriminar cada personaje de forma univoca realizando todas las preguntas posibles). Analizando los codigos de ceros y unos asociado a los rasgos de los personajes podemos deducir si es minimal o no. Para ello miramos si para una pregunta todos los personajes tienen la misma respuesta (todos S (1) o todos No (0)). En tal caso esa pregunta no es necesaria y por lo tanto el conjunto de partida no es minimal. El alumno creará por lo tanto el programa test que se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  minimal  datos/conjunto_no_suficiente.txt
```

Como salida de esta ejecución obtendríamos:

La pregunta ¿Es calvo? no discrimina a los personajes

La pregunta ¿Tiene el pelo Marron? no discrimina a los personajes

El conjunto no es minimal

3.4. Ficheros

3.4.1. Fichero de Configuración

El fichero de configuración se compone de tres partes:

1. El conjunto de preguntas
2. El conjunto de personajes
3. El nombre del fichero donde está la información para reconstruir el árbol general de rasgos.

```
#Preguntas posibles
17 #numero de preguntas
¿Tiene barba?
¿Tiene bigote?
¿Es calvo?
¿Es Claro su color de ojos?
¿Es Oscuro su color de ojos?
¿Tiene el pelo Negro?
¿Tiene el pelo Rubio?
¿Tiene el pelo Blanco?
¿Tiene el pelo Pelirrojo?
¿Tiene el pelo Marron?
¿Tiene la piel Clara?
¿Tiene la piel Oscura?
¿Tiene gafas?
¿Es Mujer?
¿Es Hombre?
¿Es Largo su pelo?
¿Es Corto su pelo?
#PERSONAJES
16 #numero de personajes
##############0#########################
#nombre del personaje
Alejandro
datos/caras1/alejandro.png #imagen del personaje

###########1#############################
Alfonso
datos/caras1/alfonso.png

##########2##############################
Antonio
datos/caras1/antonio.png

###########3#############################
Cristina
datos/caras1/cristina.png

###########4#############################
Gabriela
datos/caras1/gabriela.png

###########5#############################
Ivan
datos/caras1/ivan.png

###########6#############################
Javier
datos/caras1/javier.png

###########7#############################
Juan
datos/caras1/juan.png

###########8#############################
Laura
datos/caras1/laura.png

###########9#############################
Luis
datos/caras1/luis.png

###########10############################
Marco
datos/caras1/marco.png

###########11############################
Maria
datos/caras1/maria.png

###########12############################
Paco
datos/caras1/paco.png

###########13############################
Pepe
datos/caras1/pepe.png

###########14############################
Sofia
datos/caras1/sofia.png

###########15############################
Sonia
datos/caras1/sonia.png

#Informacion con el arbol describiendo a los personajes.
datos/caras1.tree
```

3.4.2. Fichero tree

Los ficheros con extensión tree describe un árbol general en disco. Para ellos los nodos se recorren en preorden y si el nodo existe se pone el carácter 'n' y la etiqueta correspondiente. Si no existe el nodo se pone el carácter 'x'. Así por ejemplo para la figura 3 :

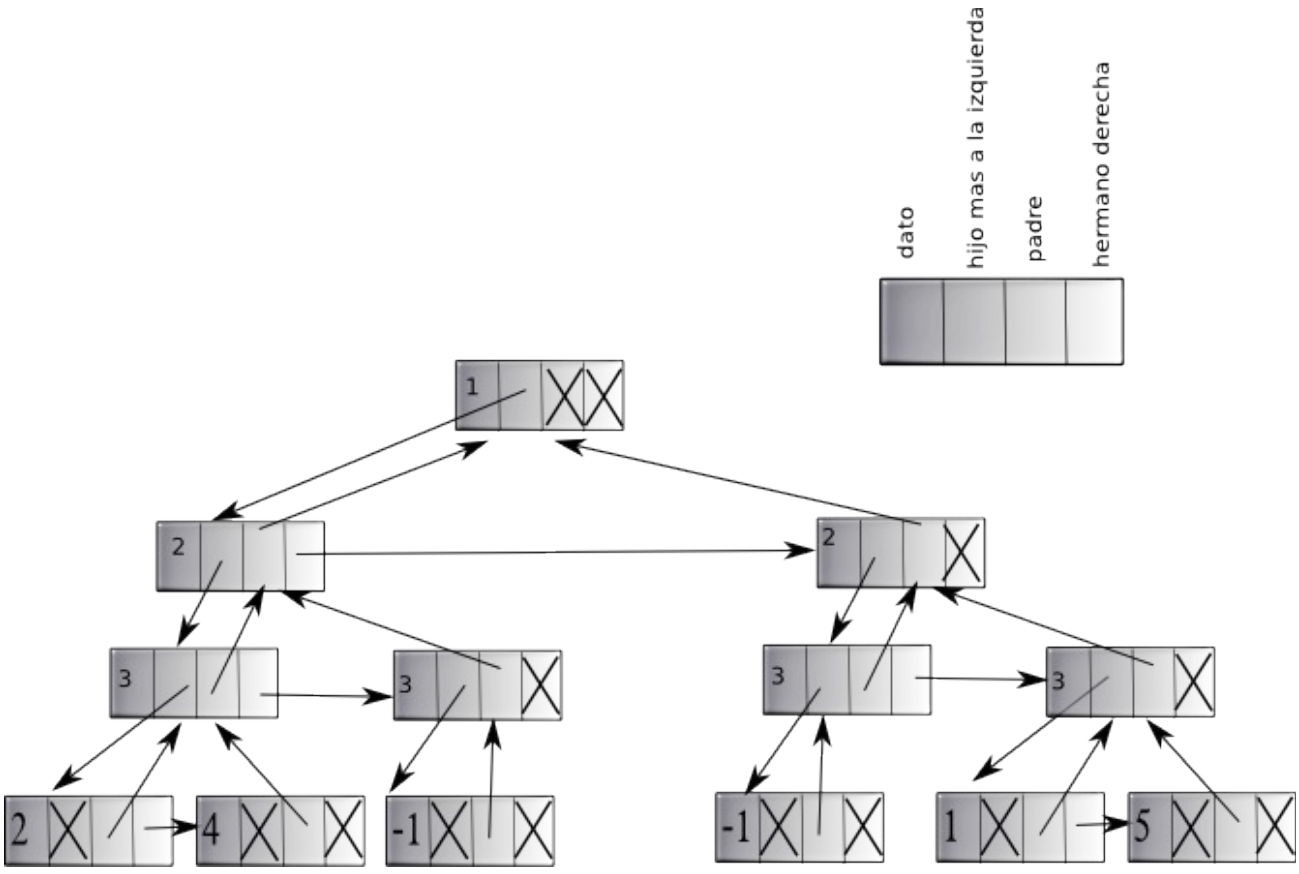


Figura 3: Ejemplo de árbol general. Donde la etiqueta en los nodos interiores son los códigos de las preguntas y en las hojas tenemos -1 indicando que no existe personaje con tal rasgo o en caso contrario el código del personaje.

El fichero tree asociado tendría en disco la siguiente secuencia:

n1n2n3n2xn4xxn3n-1xxxn2n3n-1xxn3n1xn5xxxxx

El hijo mas a la izquierda representa contestar Si a la pregunta en la etiqueta del padre. Y el hijo a la derecha (unico cuando son nodos interiores) contestar No a la pregunta del padre.

4. Módulos a desarrollar

Módulo Arbol General

Para crear los programas propuestos hará falta desarrollar el T.D.A ArbolGeneral. Este módulo debe ser lo suficiente flexible para poder desarrollar los programas propuestos. La interfaz y representación propuesta para ArbolGeneral la podéis encontrar en el fichero ArbolGeneral.h. El alumno implementará los métodos propuestos de acuerdo a la especificación dada. El T.D.A ArbolGeneral implementa un árbol en el que cada nodo puede tener un numero indeterminado de hijos. La información que almacena un nodo es:

- El elemento de tipo Tbase (plantilla)
- Un puntero al hijo más a la izquierda
- Un puntero al padre
- Un puntero al hermano a la derecha.

Con esta definición de nodo el T.D.A ArbolGeneral se puede representar como un puntero al nodo raiz. Por ejemplo un objeto de tipo ArbolGeneral donde la etiquetas sería enteros es el que se muestra en la figura 3.

Con respecto a las operaciones que se detallan, cabe destacar la posibilidad que el alumno implemente dentro de la clase ArbolGeneral un iterador que permita recorrer la clase en preorden.

```
#ifndef __ArbolGeneral_h__
#define __ArbolGeneral_h__
template <class Tbase>
class ArbolGeneral{
private:
….
public:
    ...
   class iter_preorden{
       private:
        Nodo it;
        Nodo raiz;
        int level; //altura del nodo it dentro del arbol  con raiz “raiz”
       public:
        iter_preorden();

        Tbase & operator*();

        int getlevel();

        iter_preorden & operator ++();

        bool operator == (const iter_preorden &i);

        bool operator != (const iter_preorden &i);

        friend class ArbolGeneral;
    };
```

```
    iter_preorden begin();

    iter_preorden end();

};
#endif
```

El método begin inicializa un iter_preorden para que apunte a la raiz (el primer nodo en recorrido preorden). La función end inicializa un iter_preorden para que apunte al nodo nulo.

Módulo Preguntas

El T.D.A Preguntas almacena el conjunto de preguntas dadas en el fichero de configuración.

Sería interesante definir sobre este tipo de dato un iterador que permita recorrer todas la preguntas.

Módulo Personaje y conjunto de Personajes.

El TDA Personaje mantiene la información de un personaje en memoria. Esta información es: su nombre y nombre del fichero en disco donde está su imagen. Además tendrá un código, compuesto de ceros y unos, que define el conjunto de rasgos que tiene.

Para almacenar todos los personajes definiremos un TDA que nos permita contener todos los personajes. Al igual que en preguntas sería interesante que este módulo tenga un iterador para poder recorrer todos los personajes.

Módulo Jugador

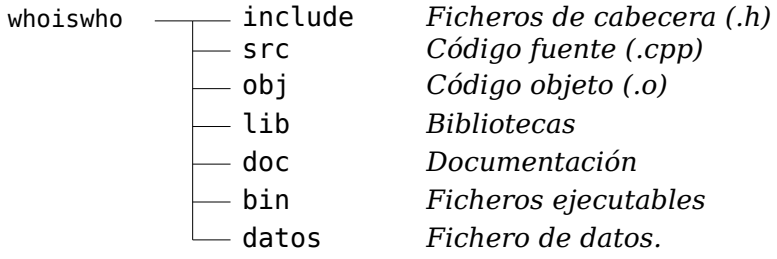
El TDA Jugador mantendrá la información del jugador: nombre, preguntas hechas, personaje oculto, etc.

En esta clase será interesante crear una o varias funciones que permitan elegir a la máquina su pregunta siguiendo las tácticas comentadas: aleatorio, maximizar la entropía.

5. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "whoiswho.*tgz*" y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*whoiswho*") para ejecutar:

```
prompt% tar zcv whoiswho.tgz whoiswho
```

tras lo cual, dispondrá de un nuevo archivo whoiswho.tgz que contiene la carpeta whoiswho así como todas las carpetas y archivos que cuelgan de ella.

6. Referencias

- [GAR06b] Garrido, A. Fdez-Valdivia, J. *"Abstracción y estructuras de datos en C++"*. Delta publicaciones, 2006.