**ÁRBOL DE DECISIONES:**

**Explicación funcionamiento del programa:**

Este programa es capaz de, a partir de la lectura de un fichero, generar un “árbol de decisiones” que permita en encontrar un elemento. El árbol está compuesto por lo que llamaremos nodos, entre los que podemos distinguir dos tipos. Los primeros son los nodos pregunta, que como su propio nombre indica, cada uno de ellos almacenará una pregunta que el programa hará al usuario. Según lo que responda, el programa avanzará por una rama u otra del árbol y ejecutará el siguiente nodo pregunta. Este proceso se repetirá sucesivamente hasta llegar al final de una de las ramas, donde nos encontramos el otro tipo de nodos, los nodos respuesta. El programa es capaz de detectar estos nodos como el final del árbol, y los ejecutará como respuesta. Sin embargo, te da la opción de cambiar la respuesta que te ha dado si no era la que esperabas. Para ello te pide que diferencies ambas respuestas a través de una nueva pregunta. Luego tenemos un nuevo nodo pregunta y un nuevo nodo respuesta que el programa se encargará de sobrescribir en el fichero de manera adecuada para que cuando se vuelva a crear el árbol, éste haya sido actualizado con los nuevos cambios.

En conclusión, este programa permite realizar la búsqueda de un elemento que ha sido clasificado en un árbol a partir de diferentes preguntas. Pero también permite clasificar nuevos elementos dentro del árbol mediante la inclusión también de una nueva pregunta.

Como implementaciones opcionales se han añadido además:

* La posibilidad de elegir previamente entre varios ficheros disponibles
* La opción de multirrespuesta generando así un árbol multicamino en vez de un árbol binario

**Estructuras de datos:**

|  |
| --- |
| Class Node |
| -ans: string  -v :vector<Node\*>\* |
| +getAns: string  +setAns: void  +getVector: vector<Node\*>\*  +allocate: void |

Node es la clase correspondiente a cada uno de los nodos existentes. La variable “ans” determina o bien una pregunta con las opciones separadas por “$” o una de las soluciones posibles. La variable “v” es un puntero a un vector de punteros. Esto es así para que su tamaño pueda ser escogido sin restricciones, cumpliendo así los tres requisitos opcionales. Cada uno de los punteros apunta a otro nodo con otra pregunta/solución, y el tamaño del vector queda determinado por el número de opciones de la pregunta “ans”.

**Descripción general del funcionamiento:**

* ***ready()*** muestra las instrucciones del programa.
* ***choose\_file()*** pregunta al usuario con que fichero se va a utilizar.
* ***Deserialize()*** se encarga de construir el árbol a partir de la lectura del fichero. Funciona de manera recursiva
* ***Serialize()*** escribe nuevamente el fichero a partir del árbol, y se utiliza por tanto para actualizar el fichero cuando se ha modificado el árbol. También funciona de manera recursiva
* ***question()*** se encarga de navegar el árbol a través de las respuestas del usuario, haciendo de nuevo uso de la recursividad. Cuando detecta que un nodo devuelve una solución en vez de una respuesta llama a la función **finalquestion()**, que pregunta si la solución es correcta.
* ***updatetree()*** es llamada en caso de que la respuesta no sea correcta, y se encarga de almacenar nuevas soluciones que son determinadas por el usuario

**Guía de uso del programa ejecutable:**

El programa ofrece tres categorías por defecto para jugar (Animales, Comida, Música) así como una opción en blanco para que el usuario cree su propia categoría(Custom). Tras escoger una categoría el usuario deberá contestar a las preguntas que se vayan haciendo con una respuesta entre las que se propongan.

Para responder utilice los números identificativos que aparecen delante de cada respuesta. Si se equivoca utilizando un número que no se encuentra entre las opciones disponibles, el programa volverá a ejecutar la misma pregunta esperando una respuesta válida. Otras preguntas que hace el programa que no son las de los nodos pregunta solo se responderán con [y/n] para *si* o *no* respectivamente.

Cuando el programa crea que tiene la respuesta correcta la mostrará por pantalla, preguntando si la solución es correcta, a lo que se debe responder con “y” para si y “n” para no. Si la solución era correcta el programa preguntará al usuario si quiere volver a jugar. De lo contrario le pedirá al usuario que le ayude a clasificar la respuesta que estaba buscando

La inclusión de los nuevos nodos es la parte mas compleja. Primero se le pedirá el nombre del elemento que estaba buscando y que el programa ha sido incapaz de encontrar. Seguidamente deberá escribir una nueva pregunta que diferencie el elemento que usted buscaba del que le propuso el programa. Por último, se le pide que diga el numero de respuestas que puede tener esa pregunta, asociando a cada respuesta un elemento. Obviamente, si selecciona un numero de respuestas k tal que k>2 (multirespuesta) se le solicitará que incluya (k-2) nuevos elementos para que todas las respuestas tengas un elemento asociado.

**Descripción del reparto de roles del equipo:**

* **Aléx Pérez Vicente:** búsqueda de información para resolver el problema de como debía escribirse el fichero para que el programa construyese el árbol correctamente.
* **Alex Pérez Vicente y Guillermo Moreno Vázquez:** escritura del programa completo como árbol binario con el respectivo diseño de algoritmos y funciones necesarias así como la escritura de algún fichero de prueba.
* **Guillermo Moreno Vázquez:** implementación opcional de una nueva función que permita la elección de un fichero entre varios disponibles en vez de utilizar un fichero único.
* **Jaime de Castro de Peñaranda:** optimización del programa, depuración e implementación opcional de multirespuesta convirtiendo el árbol en uno multicamino.
* **Todos:** Documentación del trabajo

**Propuestas de mejora y valoración personal.**

* Uno de los inconvenientes del programa es que no tiene capacidad para reestructurar el orden de las preguntas. Dada un pregunta, no se puede poner una superior que la contenga o cambiar las respuestas existentes a la misma. Tan solo se pueden añadir preguntas adicionales que especifiquen a las respuestas de las preguntas ya existentes. El usuario puede añadir preguntas manualmente cambiando el fichero de preguntas, pero sería conveniente poder hacer esto a través del programa.
* El archivo “custom” para que el usuario cree el árbol desde cero tiene un nodo inicial (“algo”) el cual no se puede eliminar con el programa actual, es decir, el árbol lo ira arrastrando por sus ramas. Habría que implementar una nueva función que permitiese cambiar este nodo inicial o alguna implementación similar.