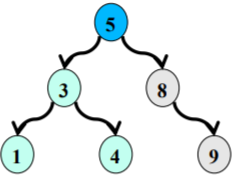
**TP Etude De Cas**

* Pourquoi un arbre binaire est-il une structure de données adaptée pour ce jeu ? Pourquoi est-il ordonné ?

L’arbre binaire est adapté pour ce jeu puisque chaque nœud de l’arbre est suivi de 2 branches comportant soit la réponse Oui soit la réponse Non. Ainsi, l’arbre binaire est optimal pour ce jeu puisqu’il permet d’orienter le joueur va l’animal qu’il a choisi.

De plus c’est un arbre binaire ordonné car il répond à ces 2 conditions :

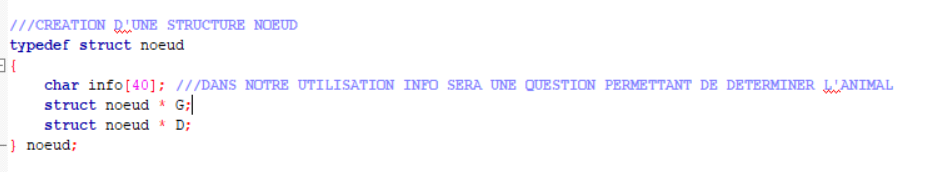
* Tout élément à gauche de la racine est inférieur à la valeur de la racine
* Tout élément à droite de la racine est supérieur à la valeur de la racine.

Cette propriété doit être vérifiée récursivement à tous les niveaux pour que l'arbre binaire soit dit ordonné

* Entre le fichier et la mémoire, laquelle des deux approches ci-dessus vous semble la plus avantageuse ? De quel point de vue ? Comparez les avantages et inconvénients des 2 approches.

Dans le cas de notre jeu, le fichier est indispensable puisqu’il doit contenir l’arbre préremplie ainsi que les questions ajoutées afin sauvegarder l’avancement de l’utilisateur. L’inconvénient du fichier c’est que cette méthode peut allourdir le projet si elle n’est pas nécessaire contrairement à la mémoire qui elle est une mémoire vive c’est-à-dire qu’elle sera effacée lorsqu’on quittera la console Code Blocks.

* D’après votre approche « préférée », définir en C la structure de données (son type) d’un nœud de l’arbre. Justifiez clairement votre définition.



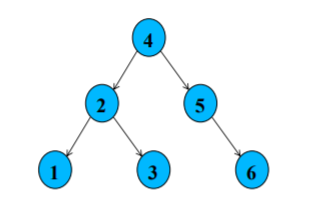
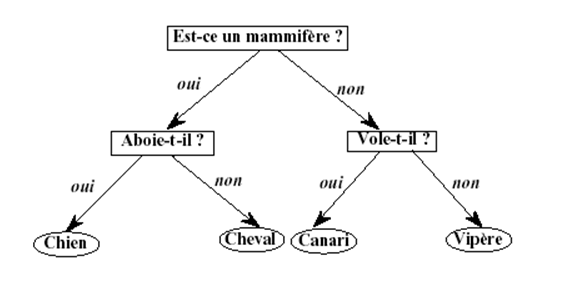
On choisit de faire une liste chainée soit une structure dans une structure. Ici, on définit la structure nœud et au sein de cette structure, nous avons 2 structures correspondant à la feuille droite et la feuille gauche juxtaposée au nœud. L’intérêt de cette méthode est de créer une seule structure de type liste chainée à 2 ancrages facilitant le parcours de l’arbre.

* D’après votre choix précédent, à quelle(s) condition(s) sait-on différencier un nœud interne d’une feuille ?

On différencie une feuille d’un nœud grâce au fait que la feuille à des pointeurs NULL sur feuille droite et feuille gauche. Ainsi, c’est le dernier élément de la chaine ce qui permet de la différencié avec un nœud puisque celui-ci garde toujours 2 possibilités de feuille.

* Quel est l’ordre de parcours (infixé, préfixé ou postfixé) de l’arbre affiché dans la trace d’exécution de la page suivante ? Justifiez. Donnez les 2 autres ordres de parcours.

L’ordre de parcours de notre arbre est infixé ainsi on parcourt celui-ci en partant de la droite. (dans la photo ci-dessous c’est un parcours infixé en partant de la droite)

C’est en comparant avec l’exemple de l’énoncé que l’on comprend que le parcours infixé sera le plus utile.