Structure : Classe

Attributs :

- Char nom[]

- Attaque attaques[]

- int points\_deplacement\_max

- int PV\_max

- int mana\_max

Structure : Effet

Attributs :

* int nb\_tours\_restants
* int valeur\_degats\_duree
* boolean paralysie

Structure : Personnage

Attributs :

* Char nom[]
* Classe classe
* int points\_deplacement
* int PV
* int mana
* Case \*position
* List Effet effets\_subis

Structure : Attaque

Attributs :

- int dégâts

- int portée

- int mana

- List Effet effets

Structure : type\_terrain

Attributs :

- Char nom[]

- boolean franchissable

- boolean franchissable\_sort

- int PD\_requis

Structure : Case

Attributs :

- int coord\_x

- int coord\_y

- type\_terrain \*terrain

- boolean occupee

Type : Case carte[]

Structure : Joueur

Attributs :

- Char nomJoueur[]

- Personnage \* groupeActuel[]

// A ajouter ? : Arbre de Compétences, nombre de parties jouées et Bonus Permanents.

Structure : nodeJoueur

Attributs :

- Joueur \* j

- Joueur \* next

Structure : listJoueur

Attributs :

- nodeJoueur \* first

- nodeJoueur \* current

- nodeJoueur \* last

Structure : Partie

Attributs :

- int nbTours

- joueurList \* participants

// Le but ici est de parcourir la liste de joueurs pour gérer les tours : la valeur courante de la liste représente le joueur dont c'est le tour. Terminer le tour d'un joueur reviendrait (entre autre) à faire un next sur la liste de joueurs. Pour qu'un joueur ne puisse pas jouer les personnages des adversaires durant son tour, on pourrait faire une fonction jouerPersonnage qui prend en parramètre un personnage et renvoie la même valeur d'entrée seulement si le personnage appartient au joueur dont c'est le tour. Pendant le programme, cette fonction pourrait être utilisée avant un appel à une fonction qui concerne les actions d'un personnage ou le deplacement.