Projet C++ : Dungeon Auto-Battle

L3 MIASHS info

Objectifs :

Le projet consiste à développer un programme C++ simulant **un combat automatique** entre deux équipes de Héros (Guerrier, Mage, Archer…). L'objectif pédagogique est de pratiquer l'**héritage**, le **polymorphisme**, les **méthodes virtuelles**, ainsi qu'une architecture orientée objets complète et cohérente.
Chaque joueur possède une liste de héros (6 maximum). Pour un affrontement, chaque joueur sélectionne **jusqu'à 3 héros**.

Un combat entre deux joueur se déroule automatiquement selon les règles suivantes :

1. Les deux joueurs sélectionnent 3 héros dans un ordre fixé
2. Les héros combattant de la façon suivante :
    a. Le héros le plus rapide a plus de chances de commencer.
    b. Les héros s'attaquent à tour de rôle.
    c. Si un héros tombe à 0 PV, il est considéré comme vaincu.
3. Après chaque combat, le héro gagnant regagne ses PV initiaux.
4. Le joueur qui a encore au moins un héros en vie gagne l'affrontement.

Le jeu est un **jeu en texte**.

Gestion du hasard :

Pour tout ce qui concerne le hasard, utilisez la fonction rand() qui renvoie un entier aléatoire. Il faut initialiser la graine avec srand(). Exemple :

```
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;
int main() {
    srand(time(0)); // À faire une seule fois au début du programme
    cout << rand() << endl;
    return 0;
}
```

Utilisez `rand()` et `srand(time(0))`.

Architecture orientée objets

Classe de base : `Hero`

Attributs suggérés :

- `string nom`
- `int pv`
- `int attaque`
- `int défense`
- `int vitesse`

Méthodes :

- `void afficherStats()`
- `virtual int getClasse() const = 0`
- `virtual int calculerDegats(const Hero& cible)`
- `virtual void perdrePV(int montant)`
- `virtual void effetDebutTour()` (optionnel)
- `virtual void effetFinTour()` (optionnel)

`Hero` doit être **abstraite**.

Classes dérivées

⚔️ Guerrier

- +20% dégâts quand ses PV < 30%
- dégâts constants
- stat principale défense

🔮 Mage

- ignore 30% de la défense ennemie
- dégâts plus variables
- stat principale attaque

🏹 Archer

- peut infliger un **coup critique** (dégâts ×2) avec un probabilité fixe
- petite chance d'**esquive (basée sur la vitesse)**
- stat principale vitesse

Chaque dérivée **redéfinit au minimum** :

- `getClasse()`
- `calculerDegats()`

Création aléatoire de héros

Créer une fonction non membre `Hero* genererHeroAleatoire()` :

- choisit aléatoirement entre Guerrier, Mage, Archer
- génère des stats dans `[50, 100]` en prenant en compte les stats principales

Combat entre deux héros

Créer une fonction : `int combat(Hero* h1, Hero* h2);`
Elle doit :

1. Déterminer qui commence (via la vitesse).
2. Appliquer les attaques en alternance.
3. Appliquer les effets début/fin de tour (si présents).
4. Retourner le héros vainqueur (1 ou 2).

Exemple de formule possible pour les dégâts :
`dégâts = (AttaqueAttaquant / max(1, DéfenseCible))*facteur_aléatoire*bonus_classe`

Classe `Joueur`

Attributs :

- `string nom`
- `tableau de Hero`

Méthodes :

- `ajouterHero(Hero* h)`
- `selectionnerEquipe()`
- possibilité de génération aléatoire

Combat entre deux équipes

Créer une fonction : `int combatEquipes(Joueur& j1, Joueur &j2);`
Elle fait s'affronter les héros sélectionnés **dans l'ordre du joueur**.
Le premier joueur n'ayant plus de héros vivants perd.

Bonus possibles (facultatifs)

- Nouveaux types de héros (Paladin, Assassin, Druide…)
- États (saignement, bouclier, brûlure…)
- Objets (amulette, potion, anneau…)
- Système de tournoi entre plusieurs joueurs
- Journal de combat plus détaillé

Les bonus améliorent la note mais ne sont pas obligatoires.

Travail à rendre (en binôme)

- Le code source complet (bien structuré et commenté).
- Une démonstration du programme lors de la dernière séance de TP.
- La soutenance portera sur la compréhension du code produit et du C++ en général.