Delinac Inès
Jouve Enzo

IUT Clermont Auvergne
1ere année
BUT informatique

SAE 1.02 - Comparaison d'approches algorithmiques

I/ Les fonctionnalités

Notre programme offre la possibilité de jouer une partie en choisissant un niveau prédéfini via un fichier. Nous disposons actuellement de trois fichiers : facile.txt, moyen.txt, et difficile.txt. Il est également possible de créer votre propre partie en ajoutant les monstres souhaités, avec le niveau désiré, à la vague de votre choix.

Par curiosité, vous pouvez afficher les joueurs triés par pseudo, ce qui vous permet, par exemple, de retrouver le pseudo d'un ami.

Pour un esprit de compétition, vous pouvez également afficher les joueurs triés par score (du meilleur au moins bon) afin de comparer votre niveau avec celui des autres.

Enfin, pour obtenir plus de détails sur les différentes parties jouées précédemment, vous pouvez afficher les statistiques d'un joueur précis en entrant son nom. Ces statistiques incluent : son nombre de scores, son meilleur score, son moins bon score, et son score moyen.

II/ Tests

Test de la fonction ajouterScore :

```
Ajout des scores
ajouter 123, 36, 98, 157

Affichage
Nom : meow
Score 1 : 157
Score 2 : 123
Score 3 : 98
Score 4 : 36

Libération des scores
```

Test de la fonction chargeChevaliers et sauvegarderTableauChevaliers :

```
Affichage du tableau chargé
Chevalier 1 : Toto
Chevalier 2 : Pilipata
Chevalier 3 : Pascale
Chevalier 4 : Amara
Chevalier 5 : Chloe
Libération du tableau après sauvegarde

Affichage du tableau chargé
Chevalier 1 : Toto
Chevalier 2 : Pilipata
```

```
Chevalier 3 : Pascale
Chevalier 4 : Amara
Chevalier 5 : Chloe
```

Test de la fonction rechercheDicho

```
1 : index 0
Statistiques pour Amara :
- Nombre de scores : 5
- Meilleur score : 60
- Moins bon score : 0
- Score moyen : 30.80
```

Test de l'aléatoire

```
Nombre aléatoire entre 0 et 3 : 2
Nombre aléatoire entre 0 et 3 : 0
Nombre aléatoire entre 0 et 3 : 3
Nombre aléatoire entre 0 et 3 : 2
Nombre aléatoire entre 0 et 3 : 2
```

Test de la fonction creerMonstres

```
Monstre : Chouin-Chouin (Niveau 1)
- Points de Vie : 4
- Points de D : 1
- Nombre d'armes : 4
Monstre : Lutin Cordial (Niveau 2)
- Points de Vie : 6
- Points de D : 1
- Nombre d'armes : 3
Monstre : Chimere (Niveau 3)
- Points de Vie : 4
- Points de D : 2
- Nombre d'armes : 5
```

Test de la fonction lireMonstres

```
Chouin-Chouin (Niveau 1) ajouter à la Vague 1.
Lutin Cordial (Niveau 2) ajouter à la Vague 1.
Chimère (Niveau 1) ajouter à la Vague 2.
Petite Pustule (Niveau 2) ajouter à la Vague 2.
Dragonus (Niveau 2) ajouter à la Vague 2.
Lecture des monstres depuis le fichier 'facile.txt' terminée.
Monstre : Lutin Cordial (Niveau 2)
```

- Points de Vie : 6
- Points de Dégâts : 1
- Nombre d'armes : 3

Monstre : Chouin-Chouin (Niveau 1)
- Points de Vie : 4
- Points de Dégâts : 1
- Nombre d'armes : 4

Monstre : Chimère (Niveau 1)
- Points de Vie : 4
- Points de Dégâts : 1
- Nombre d'armes : 4

Monstre : Petite Pustule (Niveau 2)
- Points de Vie : 6
- Points de Dégâts : 1
- Nombre d'armes : 3

Monstre : Dragonus (Niveau 2)
- Points de Vie : 6
- Points de Dégâts : 1
- Nombre d'armes : 3

tout bon !

III/ La structure des fichiers

Les fichiers sont rangés dans le répertoire /src/fichier.

Nous avons deux type de fichier :

- Le fichier scores.txt contenant les joueurs, leur nombres de score et scores

```
nombres joueurs
nom  nombre_de_score
listes_de_nombre de score
                    . . .
```

- Les fichier de jeu constitué de contexte, nombre de monstre par vague et monstre (nom, niveau) :

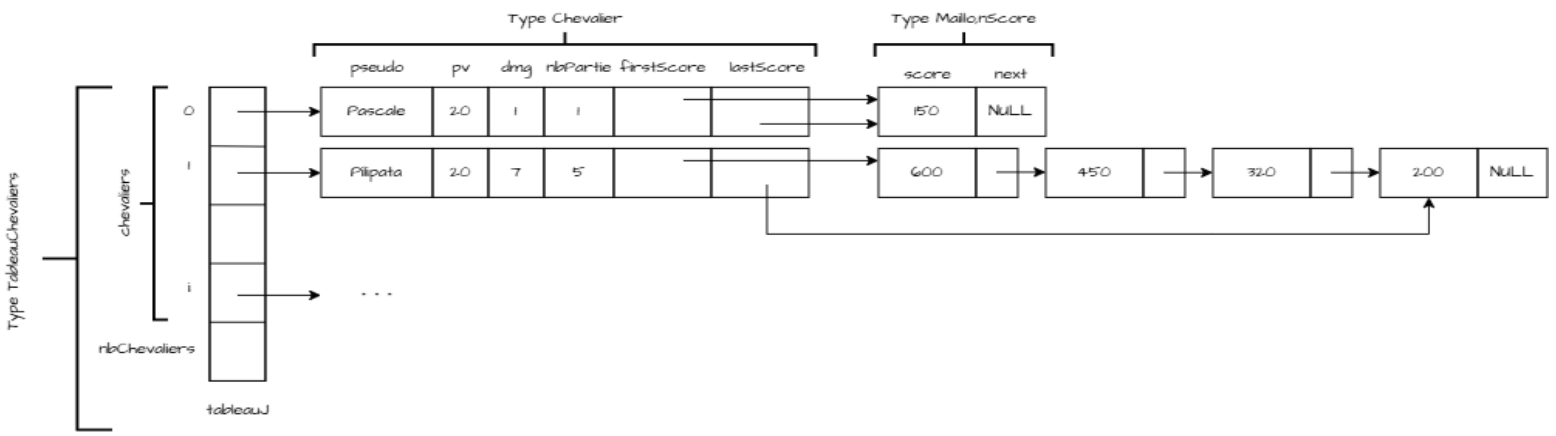
```
contexte1
nombre_de_monstre_vague1
nom_du_monstre   niveau
                    . . .
contexte2
nombre_de_monstre_vague2
nom_du_monstre   niveau
                    . . .
```

Nous avons choisis des fichiers texte pour qu'il puisse être lisible pour tous, surtout pour le fichier de partie de jeu, pour que tout le monde puisse réaliser sa propre partie (même s'il est toujours possible de créer sa partie depuis le menu).

IV/ Les structures des données

A - Les structures

Nous avons en tout 6 structures pour mieux organiser le jeu.

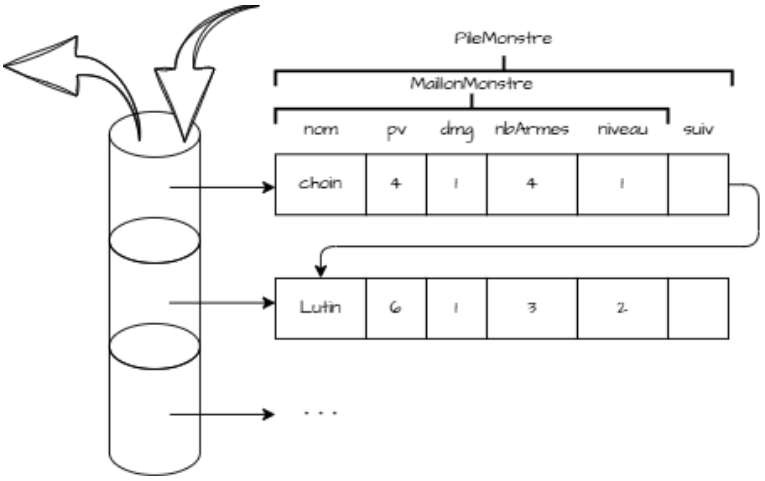


La structure MaillonScore servant à l'implémentation d'une liste chaînée de score de joueur dans sa structure avec un pointeur sur le premier score et le dernier pour un accès au donnée plus rapide notamment pour l'affichage des statistiques d'un joueur.

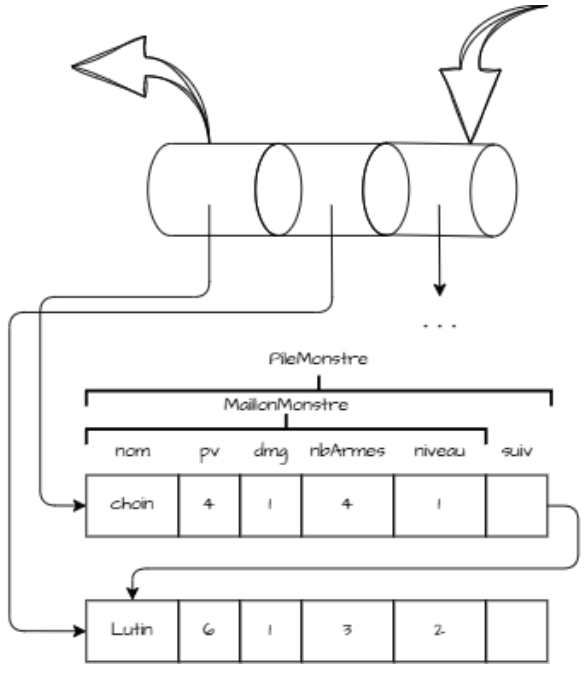
En lien avec les score, la structure chevalier contient les informations d'un joueur (pseudo, point de vie, dégâts infligés, nombre de partie joué en lien avec les pointeurs vers des MaillonScore du meilleur score et du moins bon), ce que rend l'accès au information simple est structuré.

Néanmoins, plusieurs joueurs sont enregistrés sur un fichier qui doit être récupéré pour cela, nous avons une structure de tableau dynamique avec sa taille logique, ce qui rend la recherche dichotomique efficace.

Alors, une structure Monstre est nécessaire pour faire face au chevalier tout en étant bien organisée, elle est ainsi constituée de son nom de monstre, ses points de vie, ses dégâts qu'il inflige, son nombre d'armes et son niveau (de 1 à 3, 3 étant le plus difficile à affronter).



Ainsi, pour la première vague de monstres, cela se déroule un par un alors quoi de mieux qu'une structure pile pour organiser cela.



Mais pour la deuxième vague de monstres, ils font une attaque chacun alors ce qui correspond à une structure file.

V/ Comparaison des algorithmes

A - Les algorithmes de tri

Nous avons deux fonctions de tri, ainsi qu'un tri par insertion :

- Tri rapide

  Nous avons implémenté un tri rapide, ou quicksort en anglais. Ce tri est utilisé pour organiser le tableau des chevaliers en fonction de leurs scores, dans l'ordre décroissant. Cela permet, grâce à une fonction d'affichage, de visualiser l'ensemble des chevaliers triés du meilleur au moins bon score. Grâce à sa complexité moyenne de O(n*log(n)), il est très efficace pour trier de grandes quantités de données.

- Tri A Bulle

  Un autre tri que nous avons codé est le tri à bulle. Il est également appliqué au tableau des chevaliers, mais cette fois pour les trier par ordre alphabétique en fonction du pseudo de nos preux chevaliers. Bien qu'il ait une complexité de O(n**2) dans le pire des cas, il reste adapté à notre échelle, où le nombre de chevaliers reste raisonnable.

- Tri insertion

  Enfin, notre tri par insertion est implémenté via la fonction ajouterScores, qui ajoute les scores au joueur correspondant du tableau des chevaliers dans l'ordre croissant. Il a également une complexité de O(n**2), ce qui n'est pas dérangeant pour nos petits fichiers.

B - Les algorithmes de recherches

Nous avons développé deux méthodes de recherche.

- Recherche parcours

  La recherche normale parcourt un par un le tableau des chevaliers pour trouver l'élément recherché. Bien que simple, cette méthode peut être lente pour de grandes quantités de données mais à notre échelle cette méthode ne pose aucuns problèmes. Elle est aussi avantageuse car le tableau ne doit pas obligatoirement être trié. La complexité est de O(n), donc absolument pas dérangeante à notre échelle mais peut devenir compliquée en cas d'une ouverture mondiale.

- Recherche dichotomique

  A l'inverse de la recherche parcours, la recherche dichotomique nécessite obligatoirement un tableau trié. Elle divise le tableau en deux à chaque étape, ce qui réduit ainsi le temps de recherche. Cette méthode est particulièrement efficace pour localiser rapidement un chevalier à partir de son pseudo, mais on ne ressent pas forcément la différence à notre échelle.