Qui est le plus fort des héros Marvel ?

L'objectif du projet est de déterminer parmi une liste de héros de la saga Marvel, qui est le plus fort d'entre eux.

Le vainqueur sera le héros ayant remporté la totalité de ses combats.

Les règles du jeu :

La première génération est composée de 20 personnages avec des caractéristiques choisies arbitrairement : force, agilité, intelligence, magie/puissance de feu, résistance et style.

- Lors du premier round, on obtient 10 gagnants et 10 perdants on crée donc un « winner bracket » et un « loser bracket ».
- Le second round démarre, les joueurs en winner bracket s'affronte entre eux et pareillement pour ceux en loser bracket. On obtient 2 groupes de 5 héros.

  Parmi les perdants, le plus faible est éliminé. Pour déterminer le plus faible, on fait le total des points de caractéristiques de chaque héros et on les compare.

  On réalise ensuite une série de combats pour obtenir un seul survivant du loser bracket.

- Le dernier survivant du loser bracket est réinjecté dans le winner bracket.
  On réalise ensuite une série de combats jusqu'à obtenir le gagnant final. Il sera ensuite injecté dans la génération de combats suivantes.

Les règles de combat :

Un combat se fait en comparant une à une les caractéristiques des 2 combattants : force avec force, agilité avec agilité, etc. Si on obtient une égalité, les combattants sont départagés avec leur point de style.

Le gagnant mute, il obtient 1 point de plus sur sa caractéristique la plus faible et 1 point sur la caractéristique la plus forte de son adversaire. Il obtient également 1 point de style.

Les héros conservent leurs caractéristiques entre chaque génération.

Le but est de déterminer le héro le plus fort sur plusieurs générations : par exemple, si Iron man fini vainqueur du premier tournoi, il peut perdre au suivant mais il peut aussi gagner et continuer dans la génération suivante. **Nous pensons observer une stabilisation au bout de n générations, un héro devrait sortir du lot et remporter tous les combats.**

Détails du code :

- Main.c

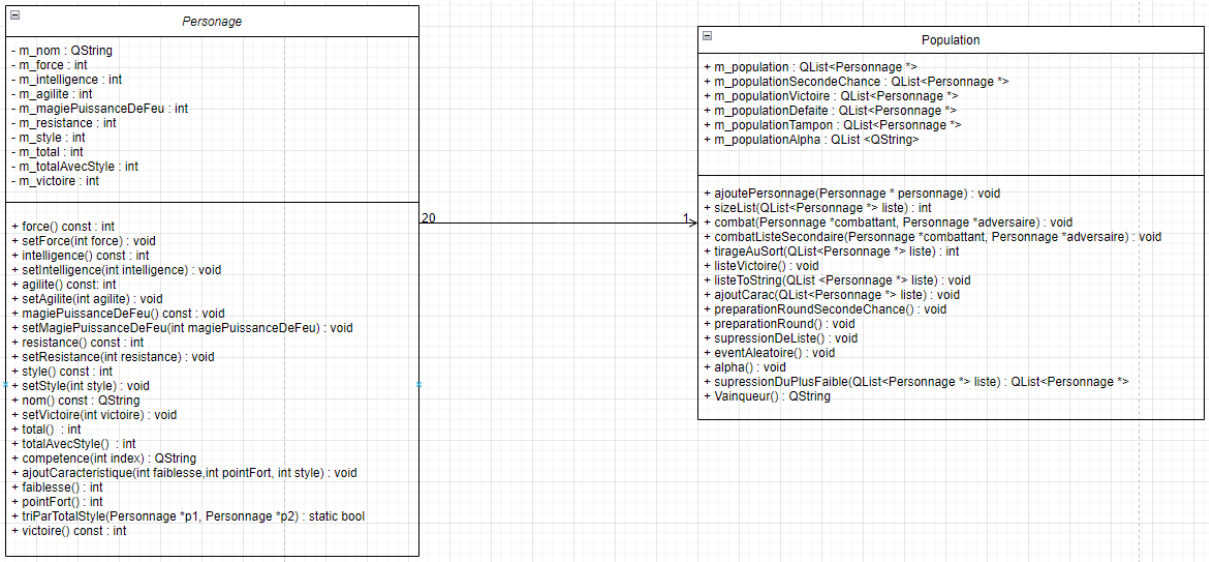
```
int main()
{
        Personnage p1("Iron Man",5,10,6,9,5,10);
        Personnage p2("Thor",10,5,5,8,8,8);
        Personnage p3("Capitaine America",6,6,6,2,5,4);
        Personnage p4("Hulk",10,6,4,4,10,7);
        Personnage p5("Black Widow",4,9,8,2,2,10);
        Personnage p6("Capitaine Marvel",7,6,7,10,6,2);
        Personnage p7("Spiderman",7,8,10,4,2,7);
        Personnage p8("Vision",4,10,8,6,3,2);
        Personnage p9("Wanda",4,7,4,10,3,6);
        Personnage p10("Black Panther",6,6,10,6,5,7);
        Personnage p11("Thanos",9,6,5,3,9,4);
        Personnage p12("Soldat de l'hiver",6,6,6,2,5,7);
        Personnage p13("Wolverine",7,4,7,0,7,5);
        Personnage p14("DeadPool",6,4,7,5,10,9);
        Personnage p15("Loki",4,10,4,8,4,7);
        Personnage p16("AntMan",4,8,6,4,6,6);
        Personnage p17("Groot",8,3,3,4,10,9);
        Personnage p18("Star Lord",5,6,5,4,2,7);
        Personnage p19("Dr. Strange",5,10,6,10,3,7);
        Personnage p20("Pr. Xavier",0,15,0,15,3,8);

        Population pp1;

        pp1.ajoutePersonnage(&p1);
        pp1.ajoutePersonnage(&p2);
        pp1.ajoutePersonnage(&p3);
        pp1.ajoutePersonnage(&p4);
        pp1.ajoutePersonnage(&p5);
        pp1.ajoutePersonnage(&p6);
        pp1.ajoutePersonnage(&p7);
        pp1.ajoutePersonnage(&p8);
        pp1.ajoutePersonnage(&p9);
        pp1.ajoutePersonnage(&p10);
        pp1.ajoutePersonnage(&p11);
        pp1.ajoutePersonnage(&p12);
        pp1.ajoutePersonnage(&p13);
        pp1.ajoutePersonnage(&p14);
        pp1.ajoutePersonnage(&p15);
        pp1.ajoutePersonnage(&p16);
        pp1.ajoutePersonnage(&p17);
        pp1.ajoutePersonnage(&p18);
        pp1.ajoutePersonnage(&p19);
        pp1.ajoutePersonnage(&p20);
```

```
    for(int i = 0; i < 2; i++)
    {
        qstd::cout<<"----------------------"<<i<<" Generation-----------------"<<"\n";
        pp1.m_populationAlpha += generation(pp1);
        pp1.listeToString(pp1.m_population);
    }


    qstd::cout<<"----------------------Stats Alpha-----------------"<<"\n";

    qstd::cout<<pp1.m_populationAlpha.size()<<"\n";

    for(int i = 0; i<pp1.m_populationAlpha.size();i++)
    {
        qstd::cout<<pp1.m_populationAlpha[i];
    }

    pp1.alpha();

    qstd::cout<<"----------------------Grand Alpha-----------------"<<"\n";
    foreach (Personnage *perso, pp1.m_population)
    {
        qstd::cout<<"----------------------------------------"<<"\n";
        qstd::cout<<" Le grand Alpha est: "<<perso->nom()<<" avec un total de "<<perso->victoire()<<" victoire\n";
        qstd::cout<<" - Force : "<<perso->force()<<"\n";
        qstd::cout<<" - Intelligence : "<<perso->intelligence()<<"\n";
        qstd::cout<<" - Agilité : "<<perso->agilite()<<"\n";
        qstd::cout<<" - Magie / Puissance de Feu : "<<perso->magiePuissanceDeFeu()<<"\n";
        qstd::cout<<" - Resistance : "<<perso->resistance()<<"\n";
        qstd::cout<<" - Style : "<<perso->style()<<"\n";
    }

return 0;
```

Dans notre main.c se trouve la déclaration de nos différents personnages et leur ajout dans notre population de base.

On procède ensuite au système de combat pour enfin afficher le grand alpha : vainqueur de toutes les générations.

- La classe personnage et la classe population, détaillée dans un diagramme UML.



- Le système de « roue biaisée », gestion des évènements aléatoires.

```
void Population::eventAleatoire()
{
    srand(time(NULL));
    int nombre= qrand() % 2;
    int nombreCarac = qrand() % 4;
    int indexPersonnage = tirageAuSort(m_population);

    if (nombre == 0)
    {
        if (nombreCarac != 0)
        {
            nombreCarac = nombreCarac - (nombreCarac*2);
        }

    }

    qstd::cout<<"--EVENT ALEA-- "<<m_population[indexPersonnage]->nom()<<" a "<<nombreCarac<<" sur toutes ces caractéristique\n";
    m_population[indexPersonnage]->setAgilite(nombreCarac);
    m_population[indexPersonnage]->setForce(nombreCarac);
    m_population[indexPersonnage]->setIntelligence(nombreCarac);
    m_population[indexPersonnage]->setMagiePuissanceDeFeu(nombreCarac);
    m_population[indexPersonnage]->setResistance(nombreCarac);
    m_population[indexPersonnage]->setStyle(nombreCarac);
}
```

Notre système de roue biaisée nous permet d'ajouter des évènements aléatoires dans nos générations. En effet, à chaque génération il y a une chance qu'un événement aléatoire se réalise et qu'un personnage aléatoire se voit ajouter ou soustraire, 1, 2 ou 3 points, à chacune de ses caractéristiques.

➔ **Pour pouvoir tester le projet, il est possible de modifier une condition dans la boucle for afin de choisir le nombre de générations souhaitées :**

```
for(int i = 0; i < 2; i++)
{
    qstd::cout<<"----------------------"<<i<<" Generation-----------------"<<"\n";
    pp1.m_populationAlpha += generation(pp1);
    pp1.listeToString(pp1.m_population);
}
```