Especificações do Código

Classes:

**Personagem:**

**Mob:**

A classe é uma interface para os mobs do jogo, com metodos virtual getters e setters para ataque maximo, ataque minimo, vida, defesa e nome.

**Npc:**

A classe é uma sub-classe de Mob, tem como atributos *float* life, max\_attack, min\_attack, defense e uma *string* para name. Possui metodos setters e getters para todos os atributos e implementa todos os metodos de Mob. Possui um construtor vazio e um que recebe todos uma *string* e 4 valores *float*.

**Boss:**

A classe é uma sub-classe de Mob, tem como atributos *life, max\_attack, min\_attack, defense* que são valores *float, name* que é uma *string. Possui também um* atributo *skill* que é da classe Habilidade. Os metodos de Mob são implementados e possui outros dois metodos que retornam valores relacionados a *skill e mais uma para settear*.

**Arma:**

**Armadura:**

**Inventário:**

**Habilidade:**

A classe tem como atributos *damage* e *spend*, valores *int*, e *name,* uma string. Tem como metodos setters e getters dos atributos. Habilidades são passadas para Personagem e Boss para serem usados nas batalhas.

Funções

**Funções Gerais:**

**Batalha:**

Biblioteca

* **int batalha(Personagem&, Mob &)**:
* **int true\_damage(int, double)**:
* **void menu2(Personagem&, Mob&)**:
* **void atacar(Personagem&, Mob&)**:
* **void atacar\_mob(Personagem&, Mob&)**:
* **void atacar\_hab(Personagem&, Mob&, Habilidade)**:
* **void atacar\_hab\_boss(Personagem&, Boss&)**:

**Treino:**

**Jogo:**

Biblioteca

* **void jogo():** A função é responsável por construir o Personagem (protagonista) usando o construtor que recebe o parâmetro “nome” (string), informado pelo usuário. Após isso, chama as fases.

* **{void Fase\_1(Personagem &player, int &t\_num):**
* **void Fase\_2(Personagem &player, int &t\_num):**
* **void Fase\_3(Personagem &player, int &t\_num):**
* **void Fase\_4(Personagem &player, int &t\_num):**
* **void Fase\_5(Personagem &player, int &t\_num):**
* **void Fase\_6(Personagem &player, int &t\_num):**
* **void Fase\_7(Personagem &player, int &t\_num)}:**

As fases (1 a 7) tem funcionamento semelhante entre elas., diferenciando apenas a história e o parâmetro “t\_num”. Cada função recebe dois parâmetros: “Personagem” (herói) e “t\_num” (número máximo de treinos permitidos na fase). Ao início da função, são construídos os inimigos e ao longo da fase a função chama os métodos responsáveis pela batalha, treino, loja, e desenvolve a história.

**Menu:**

**Loja:**

Biblioteca

* **void present\_store():**
* **void menu\_store():**
* **void seller\_random\_fail\_speech():**
* **void confirmation\_checkout():**
* **void instruction\_store(Personagem A):**
* **void remaining\_gold(Personagem A):**
* **void reading\_file(std::ifstream &file, std::vector<std::string> &archives\_name, std::vector<int> &archives\_id, std::vector<int> &archives\_price, std::vector<int> &archives\_attribute):**
* **std::vector<Arma> create\_objects\_weapon():**
* **std::vector<Armadura> create\_objects\_armor():**
* **void buy\_weapon(Personagem &A, std::vector<Arma> &weapons):**
* **void buy\_armor(Personagem &A, std::vector<Armadura> &armors):**
* **void buy\_potions(Personagem &A, int price\_on\_potions):**
* **void sell\_armor(Personagem &A, std::vector<Armadura> inventory\_armor):**
* **void sell\_weapon(Personagem &A, std::vector<Arma> inventory\_armor):**
* **void store\_weapon(Personagem &A):**
* **void store\_armor(Personagem &A):**
* **void store\_potions(Personagem &A):**
* **void store\_inventory(Personagem &A):**
* **void Funcao\_Loja(Personagem &A):**