Laboratorium 8A (10:15-11:45)

Zadanie polega na zaimplementowaniu walk między postaciami fantazy.

Dostarczony jest szkielet projektu, który jest punktem startowym do implementacji obu etapów zadania. Szkielet oprócz plików, które należy rozszerzyć, zawiera również plik *main.cpp*, który składa się z testów poszczególnych etapów i nie należy go modyfikować (oprócz odkomentowywnia kodu).

Oczekiwany output programu można znaleźć w pliku *output.txt*.

WAŻNE: W swoim rozwiązaniu należy unikać duplikacji kodu, wykorzystując do tego różne mechanizmy dziedziczenia.

Etap 0 - wprowadzenie

Wszystkie obiekty reprezentujące dowolny rodzaj postaci powinny umożliwić wywoływanie na nich następujących funkcji:

- **void Attack(Character& character)** - funkcja atakująca inną postać atakiem zadającym 2 pkt obrażeń (o ile pkt życia atakującego > 0)
- **void Rest(int hours)** - funkcja pozwalająca postaci odpocząć jednocześnie odnawiając *hours* pkt życia (pkt życia nie mogą przekroczyć maksymalnej liczby)
- **void TakeDamage(int damage)** - funkcja zadająca postaci *damage* pkt obrażeń (pkt życia nie mogą spać poniżej 0)
- **void GetHealed(int heal)** - funkcja odnawiająca postaci *heal* pkt życia (pkt życia nie mogą przekroczyć maksymalnej liczby)
- **operator<<** - wypisanie informacji o postaci (patrz output.txt)

Funkcje te mogą zmieniać swoje zachowanie w zależności od konkretnego typu postaci.

Dodatkowo funkcje te są zakomentowane w plikach nagłówkowych, ponieważ:

1. nie koniecznie zawsze wszystkie muszą być potrzebne
2. czasami może być potrzebne dodatnie do nich odpowiednich modyfikatorów

Etap 1 (4 pkt)

W pierwszym etapie należy zaimplementować klasy Character i Archer. Klasy mają już zdefiniowane wszystkie zmienne klasowe (nie można definiować dodatkowych).

Wszystkie funkcje klasy Character działają tak, jak opisano to we wprowadzeniu.

Klasa Archer rozszerza klasę Character i modyfikuje w niej zachowanie następujących funkcji:

- **void Attack(Character& character)** - Archer zamiast atakować raz robi to tyle razy ile określa zmienna *arrows_fired_at_once*
- **void TakeDamage(int damage)** - Archer otrzymuje dwa razy więcej obrażeń niż standardowa postać
- **operator<<** - oprócz standardowych informacji wypisywana jest informacja o tym, że dany obiekt jest typu Archer (patrz output.txt)

Etap 2 (6 pkt)

W drugim etapie należy zaimplementować klasy Wizard, FireballSpell i LifeDrainSpell. Klasy mają już zdefiniowane wszystkie zmienne klasowe (nie można definiować dodatkowych).

Klasy FireballSpell i LifeDrainSpell rozszerzają już przygotowaną klasę Spell reprezentującą zaklęcie. Każde zaklęcie pozwala na wywoływanie na nim następujących funkcji:

- **void Cast(Character& caster, Character& target)** - wykonuje operację rzucenia zaklęcie przez postać *caster* na postać *target*. Efekty rzucenia zaklęcia dla poszczególnych klas są następujące:
    - FireballSpell - zadaje postaci *target* 6 pkt obrażeń (użyj zmiennej statycznej)
    - LifeDrainSpell - zadaje postaci *target* 3 pkt obrażeń i leczy postać *caster* o 3 pkt życia (użyj zmiennych statycznych)
- **operator<<** - wypisuje nazwę zaklęcia (patrz output.txt) (użyj funkcji Serialize).

Klasa Wizard rozszerza klasę Character. Wizard posiada tablicę zaklęć o rozmiarze określonym przy konstrukcji obiektu (*spell_slots*). Wizard po stworzeniu nie posiada jeszcze żadnych zaklęć w tablicy (wszystkie elementy ustawione na nullptr). Wizard modyfikuje zachowanie następujących funkcji z klasy bazowej:

- **void Rest(int hours)** - oprócz standardowego efektu odpoczynku Wizard dodatkowo przygotowuje jedno zaklęcie (o ile ma na nie miejsce) i umieszcza na pierwszym wolnym miejscu swojej tablicy zaklęć. Jeżeli *hours* jest parzyste przygotowane zostaje zaklęcie FireballSpell, a w przeciwnym wypadku LifeDrainSpell.
- **void Attack(Character& character)** - zamiast atakować rzuca na swój cel ostatnio przygotowane zaklęcie i je zużywa (powinno zostać usunięte z tablicy zaklęć). Rzucenie zaklęcia nie wymaga posiadania dodatniej liczby pkt życia. Jeżeli Wizard nie posiada żadnych przygotowanych zaklęć wykonuje normalny atak.
- **operator<<** - oprócz standardowych informacji wypisywana jest informacja o tym, że dany obiekt jest typu Wizard i lista aktualnie przygotowanych zaklęć (patrz output.txt).