Софийски университет „Св. Климент Охридски“

Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране

на специалност Информатика

Летен семестър на учебната 2018/2019 година

World of Warcraft

Автор: Даниел Здравков Стоянов

ФН:45574

* 1. **Увод:**
  2. Описание и идея на проекта:

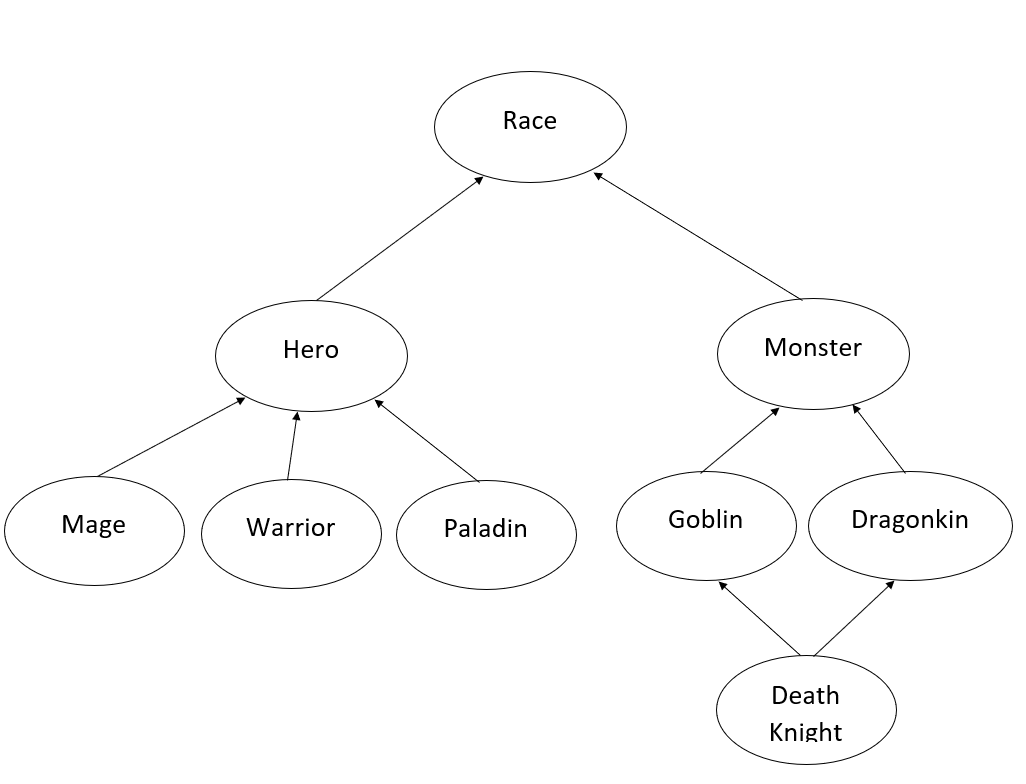
Приложението представлява игра, чиято идея е базирана на вече създадена такава, но в хиляди пъти по-опростен вариант, като се имплементира интерактивен режим, който включва множество от различни функционалности като: основно меню с главни операции, поле за регистрация на играч, влизане във вече създаден акаунт, генериране на карта с чудовища и работещ геймплей, информация за всички играчи в мрежата и техните герои.

* 1. Цел и задачи за разработка:

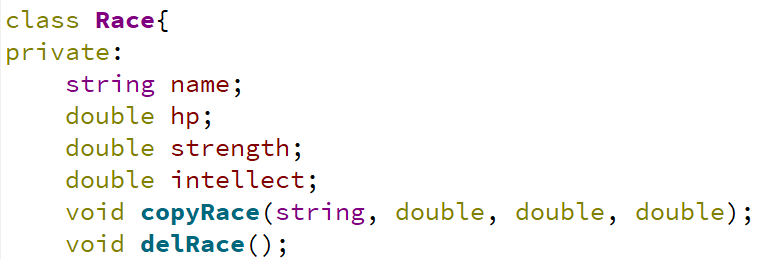
За разработката на играта бе нужен голям набор от цели и задачи, които да се имплементират, за да работи успешно приложението, а именно:

* Изграждане на подходяща архитектура и йерархия на класовете;
* Разделно компилиране на проекта;
* Използване на рационален подход при изграждането на основните функционалности и конструкции на класовете;
* Капсулиране на информацията с цел ограничаване на достъпа(външен и вътрешен);
* Реализиране на помощни класове с цел подобряване на играта;
* Управление на паметта, включително заделяне и освобождаване на динамичната памет, статични, локални и глобални променливи;
* Изработка на подходящ потребителски интерфейс, който съвпада напълно с темата на дадения проект;
* Изграждане на поле за регистрация на играч;
* Изграждане на поле за влизане във вече създаден акаунт;
* Създаване на база данни във файл с последователен достъп;
* Запазване на играчите в база данни;
* Извеждане на списъци на играчи и герои;
* Генериране на произволна карта с произволни противници, която да отговаря по най-рационалния начин на създадената йерархия;
* Валидация на входа за: избор на опция в главното меню, регистрация (ако има вече създаден такъв акаунт), при влизане във вече създаден акаунт (проверка за коректност на име и парола), въведено действие за придвижване на героя (коректност, дали се напуска картата);
* Проверка на базовите случаи за успешното завършване на играта;
* Възможност за повторно въвеждане на избор при наличието на вече грешен такъв;
* Реализиране на множество помощни функции с цел намаляване на сложността на конкретната задача;
* Статично и динамично свързване.  
    
  1. Структура на документацията:

Първо ще разгледаме архитектурата и йерархията на проекта, по-точно използването на различните свойства на класовете и представянето както на данните им, така и на примитивните им операции. След това ще покажем как операциите се свързват в различните функционалности на играта за нейното успешно реализиране.

1. **Преглед на предметната област:**
   1. . Основни дефиниции, концепции и алгоритми, които ще бъдат използвани:
      1. Клас;
      2. Конструктор, деструктор, копи-конструктор и оператор равно;
      3. Единично и множествено наследяване;
      4. Полиморфизъм;
      5. Абстрактен клас;
      6. Виртуално наследяване;
      7. Виртуални функции и чисто виртуални функции;
      8. Динамично и статично свързване;
      9. Преобразуване на типове;
      10. Файлове.
   2. Дефиниране на проблеми и сложност на поставената задача:
      1. Създаване на произволна карта с произволни противници;
      2. Четене на играчи от база данни;
      3. Проблеми с конструирането на работещо решение.
   3. Подходи, методи за решаване на проблемите:
      1. Използването на абстрактен клас даде възможността да се направи масив от указатели и чрез преобразуване на типове се постигна искания резултат;
      2. Заместване на имплементацията на база данни от двоичен файл с пряк достъп към текстов файл с последователен достъп.
   4. Потребителски и качествени изисквания:
      1. Поради обхващането на всички случаи чрез валидиране на входа от потребителите не се изисква почти нищо освен при въвеждане на число, да не се въвежда символ.
2. **Проектиране:**
   1. Обща архитектура:
      1. ****За проекта е използвана съответната архитектура на Фиг.1:

Фиг.1

* + 1. Базов клас на архитектурата е Race (Раса), който съдържа всички общи характеристики на двата си производни класа Hero (Герой) и Monster (Чудовище), които са (Фиг.2):
       - Name (име);
       - Hp (жизнени точки);
       - Strength (сила);
       - Intellect (интелект);
       - Attack() (метод за атака);
       - Defend() (метод за защита);

Фиг.2

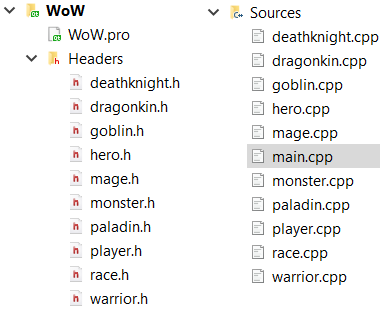
* + 1. Производният клас Hero (Герой) от своя страна е базов за трита производни класа, който има: Mage (магьосник), Warrior (войн) и Paladin (паладин). Също така Hero (Герой) е и абстрактен клас, за да не се създават инстанции от него, а само от производните му класове. Това дава възможност да се имплементира и полиморфизъм, който е необходим за изграждането на основните функционалности на героя, а това се осъществява чрез чисто виртуалните функции за атака и защита (Фиг.3):

Фиг.3

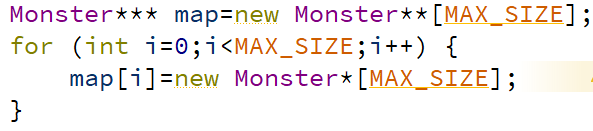
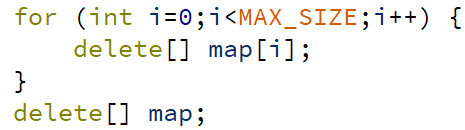
* + 1. ****Другия производен клас на Race (Раса) е Monster (Чудовище). Той има два производни класа Goblin (Гоблин) и Dragonkin, които са базови за Death Knight. Получава се така наречената „диамантена“ структура. За справянето на проблема Death Knight да наследи два пъти данните на Race, е използвано виртуално наследяване (Фиг.4 и Фиг.5):

Фиг.4

Фиг.5

1. **Рализация**:
   1. Реализиране на класовете:
      * + Класовете са реализирани чрез помощни функции за копиране на необходимите полета с цел оптимизация и съкращаване на код;
        + ****Използвано е разделно компилиране, за по-добра организация и четимост (Фиг.6).

Фиг.6

* + - * Класовете са разширени с допълнителни член данни и методи за по-добро реализиране на функционалността на играта;
      * Добавен е нов клас Player (играч), който има герой, име и парола.
    1. Управление на паметта и реализация на алгоритмите. Оптимизации:
       - Във всеки клас е реализирано канонично представяне;
       - ****Заделяне и освобождаване на динамичната памет (Фиг.7 и Фиг.8):

Фиг.7

Фиг.7

…

* + - * Оптимизациите се състоят в използването на допълнителни функции за принтиране, копиране, намиране на дължина на файл и още, запазената дума „break“ в цикли за спиране на излишни проверки, връщане на булеви резултати директно чрез return и още.
    1. Планиране, описание и създаване на тестови сценарии:
       - Интерфейсът е лесен и достъпен, не се изисква голяма предварителна информация за успешното изпълнение на играта;
       - Навсякъде има пояснителен текст за статуса на играта в текущия момент;
       - Тестов вариант е: Стартиране на играта, избиране на опция 1 за регистрация. След регистрацията се връщаме отново в основното меню и избираме опция 2 за влизане във вече създаден профил, сега можем да се избере опция 3 и започва същинската част на играта. След победа или загуба отново се намираме в основното меню, където остават още 3 опции, които са за извличане на информация относно играчите и техните герои. Опция 6 е за изход от играта.

1. **Заключение:**
   1. Обобщение на изпълнението на началните цели:
      1. Крайният продукт е работещо приложение, което отговаря на критериите на ООП и на клиента, като са изпълнени всички функционалности и дори са добавени много нови такива.
   2. Насоки в бъдещо развитие:
      1. Кода е написан за лесно негово бъдеще разширение, като може да се добавят нови класове, данни и функционалности без това да застраши изпълнението на предишните такива. Развитието може да се постигне почти във всяко едно отношение, което се дължи на голямата свобода на поставената от клиента задача.

Използвана литература: