Dungeons & Dragons

Глава 1. Увод

1.1. Описание на проекта

Проектът представлява игра, подобна на известната „Dungeons and Dragons“. Основната цел на играта е вашият герой да се придвижва по игрално поле – карта, събирайки съкровища и побеждавайки чудовища. Идеята на проекта е да се направи ролева игра, в която героят може да избере своята раса, своите предмети и с какъв вид атака да атакува (силова атака или заклинание), при битка с чудовище.

1.2. Цел и задачи на разработката

Целта на проекта е да се направи обектно-ориентирана игра, която ще съдържа карта (лабиринт), герои от различни раси и предмети, сред които различни видове оръжия, магии и брони. Задачата на разработчика е да създаде герой от избраната от играча раса, да създаде различни видове лабиринти (както по големина, така и по съдържание) и да създаде различни видове предмети, които героят на играча ще може да избере дали да прибере в инвентара си в последствие.

Играта се счита за завършена, ако героят на играча успее да премине през всички лабиринти, като преминаването започва от горния ляв ъгъл и завършва в долния десен ъгъл на лабиринта. По време на преминаването през лабиринта, играчът ще се сблъсква с чудовища, с които трябва да влезе в битка, и ще намира предмети от различни видове.

1.3 Структура на документацията

Документацията е структурирана в пет глави:

- **Увод – описание на проекта, цели и структура.**
- **Преглед на предметната област – дефиниции, концепции и методи.**
- **Проектиране – архитектура на системата и диаграми.**
- **Реализация и тестване – описание на реализацията и тестовете.**
- **Заключение – обобщение и насоки за бъдещо развитие.**

Глава 2. Преглед на предметната област

2.1. Основни дефиниции, концепции и алгоритми

- **Карта на лабиринт: Игрално поле, по което ще се движи героят на играча, като с (.) се отбелязва свободно поле, а с (#) се отбелязва стена. Също така полето съдържа поле (T) и поле (M), които сътно са за намерено съкровище и битка с чудовище.**
- **Герой: Персонажът на играча, който има жизнени показатели за сила, мана и здраве и който принадлежи към една от трите раси – човек, магьосник и войн.**
- **Съкровища и чудовища: Полета на картата, с които героят на играча взаимодейства съответно положително и отрицателно.**
- **Битка: Процесът на взаимодействие между героя и чудовище от лабиринта, който включва размяна на атаки и заклинания между враждуващите.**
- **Намиране на съкровище: Процесът, в който героят на играча избира дали иска да замени намерения предмет с някой от своите предмети.**

2.2. Дефиниране на проблеми и сложност на поставената задача.

Някой от основните проблеми включват:

- **Създаването на добре структуриран лабиринт с добре генерирани на случаен принцип обекти от тип (T) и (M).**
- **Създаване на добре балансирана система за битка между героя и чудовището.**
- **Добро управление на различните видове предмети в инвентара на героя.**
- **Правилно разпределение на бонус точките при успешно преминаване на нивото.**
- **Правилно повишаване на жизнените показатели на чудовищата в лабиринта след успешно преминато ниво.**

2.3. Подходи и методи за решаване на проблемите.

Използване на добре структуриран обектно-ориентиран код, който ще позволи разпадането на сложната задача на по-малки и по-лесни за решаване проблеми. По този начин ще се избегне генерирането на дълъг и сложен код, който предразполага към проблеми.

2.4. Потребителски изисквания и качествени изисквания

Потребителски изисквания: Играчът трябва да може да управлява своя герой, като избира какви съкровища да подбере за себе си и с какви атаки победи чудовищата. Освен това, играчът трябва да може да избира сам как да разпредели бонус точките при успешното преминаване на всяко ниво.
Качествени изисквания: Скалируемост на играта с увеличаване на нивата, поддръжка на играта и лесна актуализация на данни чрез външни файлове.

Глава 3. Проектиране

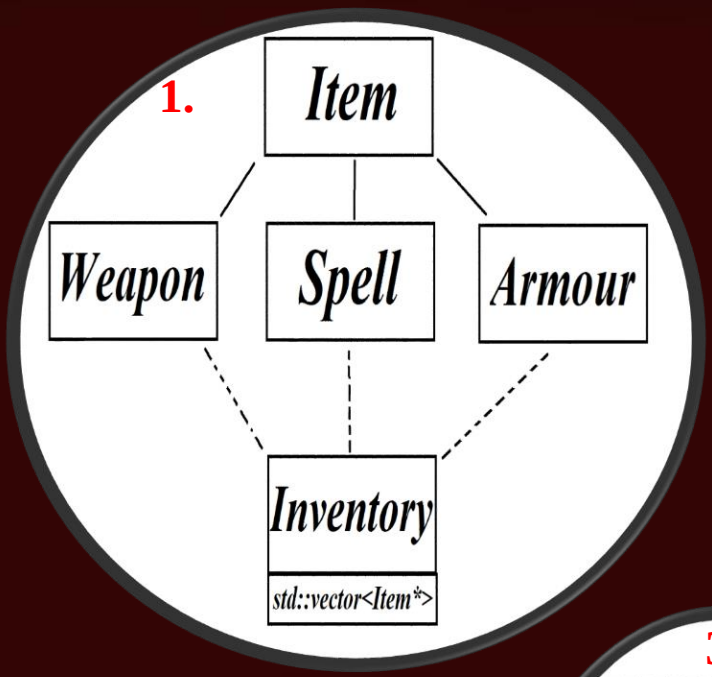
3.1. Обща архитекрура – ООП дизайн

Архитектурата на проекта включва следните основни класове:

- ***Item:***[1] **Представлява предмет, който може да бъде намерен, ако игачът попадне на поле (T). Самият предмет притежава име и параметър, който допринася към жизнените показатели на героя.**
- ***Inventory:***[1] **Представлява инвентар, който съдържа вектор от указатели към предмети. Служи за лесно управление на предметите в инвентара на героя.**
- ***Character:***[2] **Представлява героя на играча и разполага с атрибути като сила, мана и здраве, както и с инвентар, който съдържа съответните предмети на героя.**
- ***Game:***[2] **Това е класът, който отговаря за битките на героя с чудовищата в лабиринта. Също така отговаря и за паметта на героя и чудовищата. Именно тук се решава дали героят е победил чудовището, или е загубил, което би означавало край на играта.**
- ***Maze:***[3] **Това е най-важният и сложен клас, тъй като тук героят минава през лабиринта рекурсивно, събира съкровища и се бие с чудовища, опитвайки се да стигне до изхода на лабиринта.**
- ***Play:***[3] **Служи за предоставянето на различни нива, които *Maze* използва.**

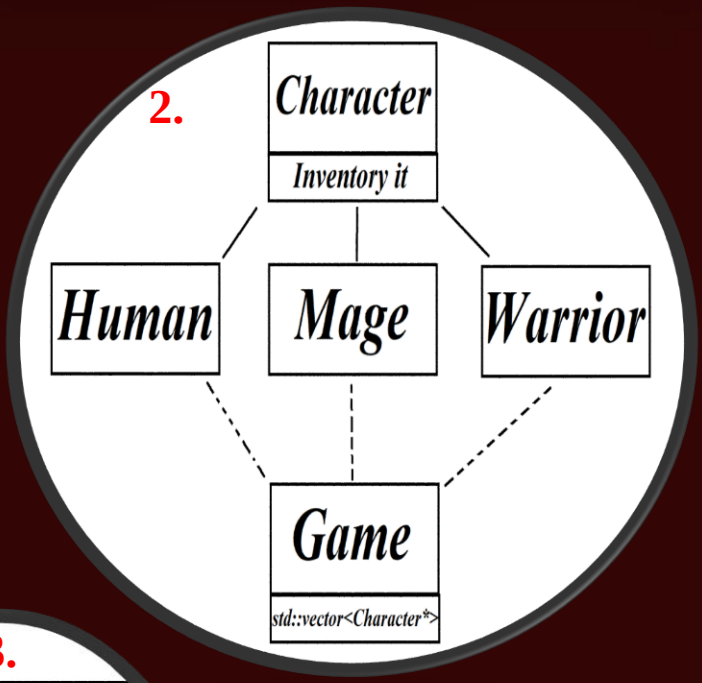
3.2. Диаграми на наследяването

Полиморфизъм на клас *Item*



Конструкция на клас *Inventory*

Полиморфизъм на клас *Character*



Конструкция на клас *Game*

Конструкция на клас *Play*

Глава 4. Реализация и тестове

4.1. Реализация на класове

Класовете Item и Character използват полиморфизъм, за да може да предоставят на класовете *Inventory*[1] и *Game*[2] лесен начин за създаване на хетерогенен контейнер (в случая *std::vector<[pointers]>*). От своя страна същите по-горе правят дълбоко копиране, за да се избегне споделяне на памет. Пример с класа Inventory:

```
Inventory::Inventory(const Inventory &other) : items()
{
    for (const auto &item : other.items)
    {
        items.push_back(item->clone());
    }
}
```

Класовете Maze и Play от своя страна използват наготово функционалностите на вече написани класове, като атрибути в своите „*private*“ променливи. Пример, съответно за класовете *Maze* и *Play*:

```
class Maze{
  public:
  ...
  private:
  Inventory it;
  Game game;
  ...
```

```
class Play{
  public:
  ...
  private:
  Maze maze
  ...
```

4.2. Управление на паметта

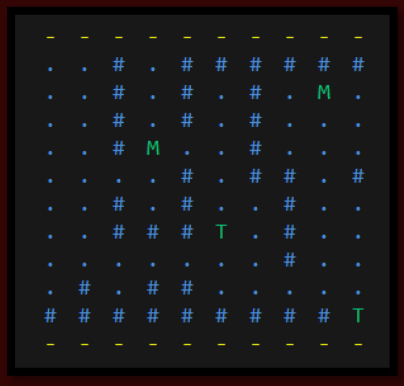
Паметта на съответните класове се управлява от съответните им деструктори. Тъй като при два от основните класове имаме полиморфизъм, то едно от нещата, който трябва да съобразим е виртуалният деструктор на основния клас. Пример, съответно за класовете Item и Character:

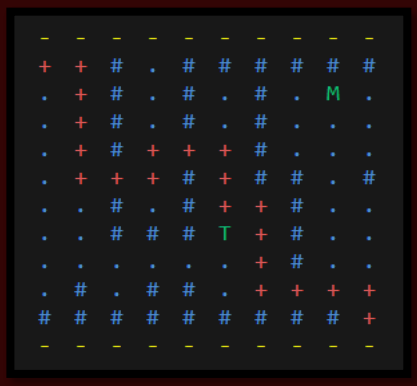
```
class Item{
  public:
  virtual ~Item() = default;
  ...
  private:
  ...
```

```
class Character{
  public:
  virtual ~Character() = default;
  ...
  private:
  ...
```

4.3. Планиране, описание и създаване на тестови сценарии

Тестовият сценарии трябва да отговаря на всички зададени условия, тоест да провери дали лабиринтът има изход, дали броят на чудовищата и броят на съкровищата отговарят на зададеното условие и дали героят на играча се движи правилно по картата. Примерен, съответно начален и преминат лабиринт:





Глава 5. Заключение

5.1. Обобщение на изпълнението на началните цели

Разработеният прототип на играта *Dungeons and Dragons* успешно реализира основните идеи за придвижване на героя в лабиринт, събиране на съкровища, битка с чудовища и успешно разпределяне на бонуса на жизнените показатели при повишаване на нивото.

5.2. Насоки за бъдещо развитие и усъвършенстване

В близкото бъдеще ще бъдат добавени много нови нива и нови функционалности като например:

- Предмети за чудовищата с напредване на нивото.
- Промени по расите на героите, за да има повече възможности за играча.
- Добавяне на функционалност, такава, че играчът да може да избере да избяга от битката и да потърси друг изход.
- Добавяне на полета (H) в лабиринта, такива, че ако героят на играча попадне на такова поле да се излекува с определен процент от здравето си.
- Добавяне на функицоналност за автоматично генерирани лабиринти, които ще отговарят на всички изисквания по-горе.
- Добавяне на функционалност, която ще позволи надграждане на даден предмет, ако бъде намерен предмет от същия тип.