**Introduction:**

This project aims to produce a program that uses object oriented programming in C++ to produce a text based adventure game in the console. The use of OOP includes classes and inheritance, polymorphism and the implementation of virtual and pure virtual functions. The aim of the game will be to pass through a set of “dungeons” picking up different weapons and fighting different enemies to reach the exit. weapons and enemies can be made using a base class and constructors to cover their basic stats and behaviours. Your character will be able to move through a map that can be printed into the console as a static picture to help the player move through the level.

**Design:**

**UML Diagrams:**

|  |
| --- |
| Character |
| -\_health : int  -\_positionX : int  -\_positionY : int  -\_name : std::string |
| +\_attack() : virtual int  +\_move() : virtual void  +\_miss() : virtual bool  +\_damage(int a, int b) : virtual int |

|  |
| --- |
| Enemy : Character |
| -\_a : int  -\_b : int  -\_missChance : double |
| +\_damage(int a, int b) : int |

|  |
| --- |
| Player : Character |
| -\_<vector> equipment : string  -\_isHealth : int |
| +\_isHealth()  +\_isEquipment()  +\_damage(int a, int b) : int |

|  |
| --- |
| Weapon |
| -\_damage : int  -\_name : string  -\_missChance : double  -\_symbol : string  -\_a : int  -\_b : int |
| +\_Weapon(string name, string symbol, int damage, double missChance)  +\_equip() : void  +\_ chooseFirst() : void  +\_getSymbol : string |

|  |
| --- |
| Map |
| -\_<vector> mapLayout : vector |
| +\_displayMap() : string |

**Function Description:**

**Character:**

**attack():** Method that calls on the damage and miss-chance of the players equipped weapon. Checks for an enemy character next to the player character and if so, deals damage by subtracting from the enemies health variable. Calls the damage function for find the exact damage value. If no character next to the player, nothing happens.

**move():** Takes input of either w, a, s or d from the keyboard and moves the player in North, West, South or East respectively given the north points towards the top of the screen. Checks if there is an object (i.e. Wall) in the way, if so, nothing happens.

**miss():** Is called every time attack is called and before damage is dealt. Takes the miss- chance of the weapon and calculates weather the attack hits or not. Since enemy characters do not carry weapons, they have their own miss-chance variable.

**damage():** used to calculate exact damage when at attack hits. Uses variable a and b from the weapon class and determines a value between a and b to be the amount of damage. This value is then subtracted from the health variable of the character. enemies have their own variables a and b, these are used since they do not use weapons.

**Player::Character:**

**isHealth():** returns the current health of player when called.

**isEquipment():** returns the vector containing all items/weapons current held.

**Weapon:**

**equip():** If an item is ‘walked over’ by the player icon, the this function is called and add the item onto the end of the equipment vector and removes I from the map.

**chooseFirst():** when called, takes the element at the head of the vector list and moves to the end of the list, making the second item in the list the new head of the list. This allows the player to choose the current item ‘in hand’.

**getSymbol():** outputs the symbol variable from the weapon object to be displayed.

**Map:**

**displayMap():** prints the map out to the ncurses window.

**createMap():** Creates the map using loops to (in this simple case) creating one room with walls, and no doors.

**Testing:**

The majority of our testing will done during run time by testing al the possible inputs and checking the outputs against expected outputs of our program. Testing subsection of our program will involve placing temporary outputs in function calls to test that are being called correctly when the game starts up and when keyboard and mouse inputs are used.

A make file has been created to assist this, having a standard compile method and a testing method allowing a string of inputs to be tested at a time and outputs to say whther the test was successful or not.