Dylan Phoutthavong
July 1st, 2024
CSC 1061
Jeff Hemmes

M3 Capstone Project: Data Structures

Abstract Data Types Used:
In my project, I have used several ADTs, including vectors and maps. I chose vectors for managing dynamic collections like the player's inventory and lists of items in rooms because of its flexibility. Maps were used for storing connections and descriptions due to the efficient key-value pairing, this allows for quick lookups.

Implementation:
I chose the implementations of vectors for its dynamic resizing capabilities and the ease with which it helps to allow for the addition and removal of elements. Maps were selected for the efficiency in accessing elements via keys, this is essential for managing complex room connections and their descriptions.

The data in my program includes:

- Room Descriptions: Text data describing each room and its connections to other rooms.
- Item Info: Details about items, such as name, descriptions, and their effects.
- Monster Details: Info about monsters, including stats and behaviors.
- Player Data: The player's current stats, inventory, and health.

Sorting Algorithm:
My program does not require sorting of data, but I have used simple algorithms to keep the player's inventory for quick access.

Inheritance:

The inheritance hierarchy in my project includes a base class '**Character**' with derived classes '**Player**' and '**Monster**'. This structure allows for shared characteristics and behaviors while enabling specific implementations for players and monsters

Program Features:

- Inheritance and Polymorphism: This code showcases the use of inheritance with the '**Character**' base class and the derived "**Player**" and '**Monster**' classes. It also demonstrates polymorphism using virtual functions, allowing different implementations of '**displayStats**' in '**Player**' and '**Monster**'
- Inventory Systems: The '**Player**' class includes vector to manage the player's inventory, showing the use of the vector structure for dynamic collections.

Feedback Implementation:
The code is now organized into separate files for better maintainability:

- Character Class:
    - Defined in '**Character.h**' and '**Character.cpp**'
    - Contains basic attributes and methos for all characters.
    - Attributes are '**protected**' to allow access in derived classes
- Player Class:
    - Defined in '**Player.h**' and '**Player.cpp**'
    - Inherits from '**Character**'
    - Manages player's inventory and current room
- Monster Class:
    - Defined in '**Monster.h**' and '**Monster.coo**'
    - Inherits from '**Character**'

This revised program supports my overall capstone project to ensure a clear, maintainable codebase and enabling an improved combat and inventory mechanics in the game.