Description:

The topic that I chose was the text-based adventure game. My program will be based around a sci-fi horror survival game, in which the user will control the player by typing commands into the terminal. The program will be used by entertainment seeking adults, age verified for the game. I believe the game will provide the user with an “escape from reality”, where the user can read through a story and choose the protagonist’s actions. The user will enter pre-defined commands to control the player and will go through an age verification process where the user can select a username for the player.

Concepts to implement:

Classes/abstraction:

This project will be heavily based on objects, both in the characters and the world around them. I suspect I will implement many cases of classes in the form of:

• NPC

• Items

• Rooms (With description)

• Users (with username and overall progression)

These topics will include levels of abstraction to keep the back end, containing user information and progressions, confidential.

Pointers/destructors:

One issue I foresee is too many objects existing within the program, which may cause the program to slow down or cause memory errors. I plan to work around this issue, with consolidation. The idea is to implement many cases of pointers and destructors. These will ensure that items that exist will not be copied, and when the player is done, items/rooms will be deleted until revisited.

Exception Handling:

Since the user will be constricted to entering commands to control the player, I will provide the player with an instruction manual, detailing the several commands they can complete. The user may accidentally enter commands with typos or with actions that are not possible with the player's specific circumstances. With this, I will provide exception handling to keep the user playing without any program exceptions being thrown.

Side note: Other concepts will be implemented later to improve the efficiency of the program once we learn about them.

Algorithm:

The general game play flow will involve a linear progression where the player will load the game, from save if applicable, go through a game intro, then start the game loop where the actions of the player will determine the rest of the logic (Located in Player State Update). Exit upon request at any time will save the game and close program. The player state update will take in keyboard actions, which will then lead to another state update which will return information that will be used to print information to player. The Item State Update, will determine what kind of item and how the player wanted to interact with it, then return information to player state. The Character State Update will determine what kind of character the player is interacting with, whether the player has already been introduced, and update dialogue based on response of player. Changes made during this update will be returned to player state. Location State Update will determine if the player is changing Sectors (World Obj array) or current world array index. Changes, or denial of changes, will be made based on the current location within the world objects array and/or the ind of the single world obj array.

Exception Handling:

For all user input, there will be a try and catch block where the user input will be protected from errors. Firstly, in the case where the user enters letters, all input will be transformed to lower case, then compared with a pre defined, action list. In the case where the action is not found, the program will throw an error where the catch statement will run a loop till the user enters a correct command. In the case where the user accidentily enters a different value, the catch statement will catch all other errors and place the user within that same loop.

Searching and Sorting:

The items in this program will all have keys. These keys will both be used to keep track of item identification, as well as interact with items in the game world. To do this successfully, the player will hold an inventory that will use quicksort to sort the players inventory and binary search to check for items already in inventory, as well as match it with surrounding items. These will only be done with the item key values, thus changing item positioning themselves.