CptS 122- Data Structures

Programming Assignment 3: Hunt the Wumpus Game (C++)

Assigned: Tuesday, May 23rd, 2020
Due: Tuesday, June 2nd, 2020 by midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in **C++**
- Declare and define *constructors*
- Declare and define *destructors*
- Compare and contrast *public* and *private* access specifiers in C++
- Describe what is an *attribute* or data member of a class
- Describe what is a *method* of a class
- Apply and implement *overloaded* functions
- Apply and implement overloaded *operators* (stream insertion and stream extraction)
- Distinguish between pass-by-*value* and pass-by-*reference*
- Discuss *classes* versus *objects*
- Apply basic *file* operations on file *streams*

II. Overview

Developed in 1972 by Gregory Yob, *Hunt the Wumpus* was one of the first text-based adventure games for computes. In the game, you wander around a maze of caves, looking for the gold and running away from the dreaded Wumpus (or possible more than one wumpi). The object of the game is to find the gold without running into the Wumpus or fall into any bottomless pits.

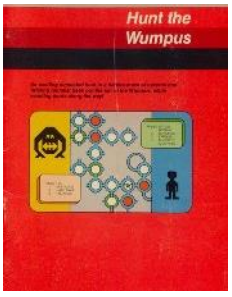

Figure 1-- Hunt the Wumpus Box Cover

III. Specifications

The game is played on a 5x5 grid. The grid contains a player (you), a wumpus (the enemy!), a pot of gold (reward) and some pits (random number from 5 - 10). Moving over a pit or onto a wumpus loses the game. Moving over the gold wins you the game.

The state of the game should be stored in a class called GameWorld. A private field of that class should be a 2-dimensional array of size 5. This array should store the state of the game. As this is a class, you should have at least the following public members:

- `GameWorld(...)` // constructor that creates the game with a random startup
- `displayEntireWorld(...)` // This should display the game state to the screen
- `displayVisibleWorld(...)` // Displays all squares one away from the player
- `moveUp(...)` // Move the player *up* one square
- `moveDown(...)` // Move the player *down* one square
- `moveRight(...)` // Move the player *right* one square
- `moveLeft(...)` // Move the player *left* one square
- `haveIWon()` // returns true or false if the player has won
- `amIAlive()` // returns true or false depending on if player hit a Wumpus or pit

IV. Tasks

Your code should allow the player to, starting from a random location, explore the game world. Start by creating an object of GameWorld in main (which should call the constructor which sets up a random instance of the world. Then display the visible world and display a menu letting the user make some moves.

This menu should take the following input:

Need to:
Play game from start with new board
Play game with same board after win/loss
Play game with new board after win/loss

- i or I should move the player up
- k or K should move the player down
- j or J should move the player to the left
- l or L should move the player to the right
- v or V should use the `displayVisibleWorld` show what is in the caves adjacent to the player
- c or C should cheat and show the entire state of the game using `displayEntireWorld` function
- r or R should restart the game with the same player
- n or N should restart the game with a new player
- q or Q should end the game

After implementing the appropriate action, your code should check to see if it is over a wumpus or a pit, or if it has won by getting the gold. Display the results. You should then loop over this

code until either the player finds the gold or dies. You should also display appropriate error messages if the player tries to move in an **invalid direction (tries to leave the board)** or uses an **invalid input.**

Points Awarded:

- For every move the Player stays alive: 5 points
- For every time the Player uses `displayVisibleWorld`: - 2 points
- For every time the Player `displayEntireWorld`: - 5 points

Points should be tabulated and then saved in a file called "`GameScores.txt`" with Player's Name and corresponding points. (You must utilize an overloading operator

V. Console Output Example:

- W = Wumpus
- P = Pitt
- U = Player/User
- G = Gold

			P	
P		U		P
	W			G
P			P	

Figure 2—displayEntireWorld( ) Function Call

	1 1	1 2	1 P 3	
	2 1	2 U 2	2 3	
	3 W 1	3 2	3 3	

Figure 3 – DisplayVisibleWorld( ) Function Call

VI. Submitting Assignment:

1. Must submit your assignment in a zip file through ***blackboard***.

2. Your project must contain at least one header files (.h files) and two C++ source files (which must be .cpp files). There should be one .h file per class declaration. There should be one .cpp for each set of operations belonging to a single class and one for the main () function.

3. Your project must build properly. The most points an assignment can receive if it does not build properly is 65 out of 100.

- 50 points – GameWorld Class implementation (5 points each)
    1. `GameWorld`
    2. `displayEntireWorld`
    3. `displayVisibleWorld`
    4. `moveUp`
    5. `moveDown`
    6. `moveRight`
    7. `moveLeft`
    8. `haveIWon`
    9. `amIAlive`
    10. `Others??`
- 5 points – Clean console output of the Wumpus Cave Grid
- 30 points – Calculations of the player points based on the points system described above
- 5 points – Opening and Closing a File
- 5 points – Input Validation (character input and array out of bounds)
- 5 points – Appropriate top-down, style, and commenting according the class standards

		U		