TDQC

Hangman

Jack Spence

20 July 2018

1. Functionality

1.1 Requests

The request for this assignment was to create a hangman clone in the C programming language. The base requirements were to have the game read a word-list, either supplied from the user or in the current working directory with the filename “.words”. Next it would select a word from the list supplied to choose as the word to be guessed. User’s were then prompted for a single character that they suspected the hidden word to have. If they were correct, each correct character in the hidden word would be revealed. Every incorrect guess incremented their guess count, when it reached six incorrect guesses they lost.

At the loading of the game, the program was also required to check if a “.hangman” file was found, which should contain the number of wins, loses, and average number of guesses per win. This file was then updated at the games end if the player won.

1.2 Suggested Features

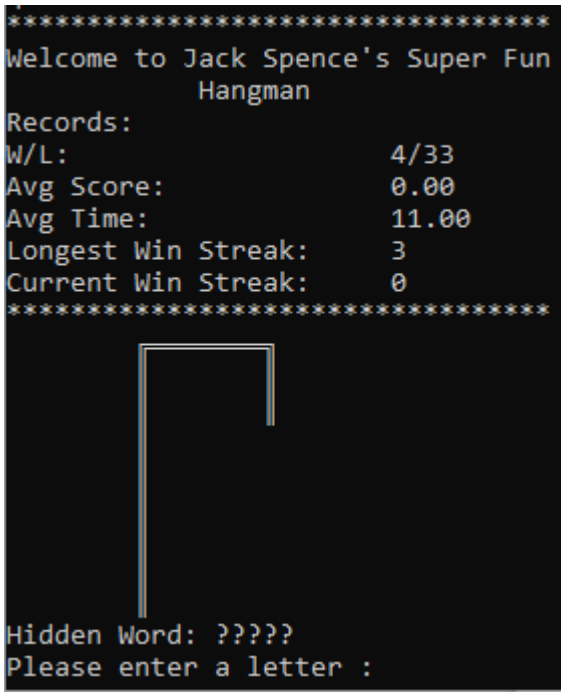
Suggested features attempted in my submission are to allow phrases that contain punctuation marks, keeping additional statistics in the “.hangman” file, and displaying the gallows and the poor hanging person as incorrect guesses are made.

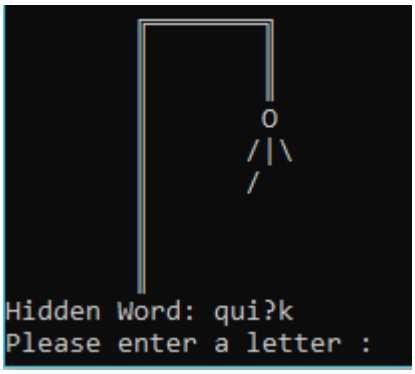
1.3 Syntax

This program does not have any switches and the only command line argument other than the program name is the word-list the player can supply.

./hangman [filename]

1.4 Sample Output





2. Project Write-Up

2.1 Initial Design Plans

The steps I initially took to handle this project was a “hit the ground running” technique. I started with the user and file input, built the main play game function, then started on getting them to work together and checking for errors as the flow goes along. The last step of the base requirements was to work with the statistics that were saved as well as saving them when the game was over.

I did the suggested features in the order I presumed to be the hardest to the easiest that I was going to do. The flow of those went from the non-alphabetical characters, into the additional statistics, and finally into the hangman display.

2.2 What didn’t work

I had attempted to do the UTF-8 suggested feature as well, but I could not quickly understand how to parse variable length characters and implement that feature without significantly more work than the other suggested features. That branch was abandoned and never pushed to Gitlab.

Another quirk I had found was with the hangman display, my original implementation was a switch statement that had each value of the number of guesses mapped out with the corresponding output inside the statement. This implementation wound up being well over 100 lines of code alone. It was rewritten into a for loop that contains several if statements that reduced the lines in half.

2.3 What went well

I was quick to be able to parse user input and get valid submissions as well as use the same function to parse file input. Being able to reuse the function for multiple purposes made development much easier in the long run.

I was also much more comfortable with using pointers and tried my best to do most of my functions as pass by reference.

2.4 Conclusion

This hangman project was one of the more intensive projects I have encountered in this course so far. I underestimated the difficulty when the requirements sheet was handed out but quickly realized it was going to be a challenge. I believe the product I have created meets the intent and satisfies all the necessary requirements.