Ra’ed Alasmar

V2Y3X8

Developer’s Documentation

Welcome to the developer's documentation for the Hangman game implemented in the C programming language. Throughout this guide, I'll walk you through the game, detailing the purpose of each function, including their parameters and outputs, and providing a closer look at how everything comes together.

Overview of the Hangman game.

Hangman is a classic word-guessing game where the user chooses the difficulty level and the length of the word. A word with the desired length is dynamically chosen and represented by dashes. The player then guesses letters, aiming to uncover the hidden word while staying within the allowed number of mistakes.

Here is a list of the functions that I used with their parameters:

Void clear_input_stream ( ):

The clear input stream function has a function type of void. The goal of this function is to clear the input stream ensuring that any unwanted characters are removed. This function also prevents issues related to white space characters.

Void start_or_end (char *level):

this function will give the user the option of starting ‘S’ or ending ‘E’ the game, the user’s input is also converted to upper-case to avoid any case-sensitive errors. If the user chooses to start the game then the **main_menu** function will be called, if the user chooses to end the game then the terminal is cleared and a farewell message is displayed, and the program exits. In the case of invalid input, the function prints an error message and clears the input stream using the **clear_input_stream ( )** function and returns to the beginning of the **start_or_end** function (using recursion) so the user can be asked again to enter a valid choice.

Void main_menu (char *level):

Introduces the variable **level choice** which is the level chosen by the user, encloses the menu in a do-while loop, welcomes the user, and explains the rules of the game. Following this, the program clears the input stream, gets the character from the user, and converts it to upper-case to avoid case-sensitive errors. A ‘**switch’** statement uses **levelchoice** to determine the maximum allowed mistakes for the chosen difficulty level and modifies the value pointed to by the ‘**level’** to store the selected difficulty level. The do-while loop also ensures that the user selects a valid difficulty level ('E', 'I', 'A', or 'M') and clears the terminal to ensure a clean user experience.
The goal of this function is to let the user choose the difficulty level that he/she would like to play.

int mistakesallowed (char level):

The goal of this function is to set the mistakes allowed by the user, it does that through getting the **level** variable as a parameter and utilizes a switch statement with the available difficulty options and returns with their number of mistakes.

Void insertdashes (char guessed [], int wordlength):

The goal of this function is to insert dashes in the guessed array to represent the unknown word. The parameters of this function are guessed array which is a visual representation of the word being guessed, where underscores represent letters that haven't been guessed yet, and correctly guessed letters replace the underscores and the word length, and it sets the last character In the string to null making it a valid string.

Char *randomword (int length):

The goal of this function is to get a random word with the desired length of the user. An infinite while loop is implemented until we achieve a word with the correct length. Checks the length of the word if it is less than 14 it will open a “sortedtext” file and if 14 and above it will open “longerwords” and get a random index depending on the number of words in the file and check if the file was opened successfully. Then, the function opens the selected file and reads lines until it reaches the randomly chosen word storing it in dynamically allocated memory for the ‘word’ variable. It checks if the last character of the word is a new line character and replaces it with a null terminator character if necessary. In the end, the program closes the file and checks if the length of the generated word matches the desired length. If it does, breaking the loop frees the allocated memory for the word and returns it.

void displaywrongcharacters (char wronglyguessed [], int count):

The goal of this function is to display the wrongly guessed characters, also it is used to check if the user repeated a mistake so it does not count as a mistake again and warn the user that the letter has been already guessed