Hangman project developer documentation

1. Libraries

- **stdio.h**
- **stdlib.h**
- **string.h**
- **time.h**

These are standard C libraries that provide functions for input/output, memory allocation, string manipulation, and time-related operations.

2. Functions

- **Struct definition:**

Defines a linked list node containing a word and a pointer to the next node.

- **File Opening Function:**

FILE* openFile() this function Opens a file named "words.txt" for reading and returns the file pointer. If the file was not opened successfully then the message (Error File could not be opened) will be printed, if the file was opened without errors, then the message (the file was opened successfully!) will be printed.

- **function to add elements to our linked list:**

void addelement this function purpose is to add a new elements at the head of the linked list , first it Allocate memory for a new Node , Check if memory allocation was successful , if it is successful then Copy the given word into the 'word' member of the new node , Set the 'next' pointer of the new node to the current first element of the linked list and finally update the head of the linked list to point to the new node .

- **function to print the linked list:**

void printList , this function purposes is to print all the elements of the linked list , it will not be used in the program it was created for the purpose of testing the code .

- **Function to free the memory allocated for the linked list:**

void freeList , the function will take a pointer to the head of a linked list as its parameter, iterate through the linked list and deallocating the memory used by all nodes in a linked list.

- **Function to choose a random word with a specified length:**

randomWordWithChosenLength , this function has parameters an integer number which is the length chosen by the user and a pointer to the head of a linked list . and first we create

a dynamically allocated array to store words of the specified length in addition a Counter to keep track of the number of words found with the specified length.

Traverse the linked list and store words of the specified length If a word of the correct length is found, dynamically allocate memory for the word and store it in wordsOfLengthRandomLength.

if no words of the specified length are found, print an error message (No words found with the specified length)

srand(time (NULL)); this function is used to Seed the random number generator using the current time. This ensures that the random sequence is different each time the program is run.

After that, Generate Random Index within the range of the number of words found.

Retrieve the randomly selected word from the array.

Finally, Free the memory allocated for the temporary array used to store words and return the randomly selected word to the calling function.

- ## get Word Length from User:

getWordLengthFromUser() , this function ensures that the user inputs a valid word length within the specified range (1 to 16). It uses a loop to repeatedly prompt the user until valid input is received, handling errors, and clearing the input buffer as needed. finally, the word length integer number will be returned.

- ## removeLetter Function:

the function takes a string randomWord and a character c as parameters. It creates a new string modifiedWord by removing all occurrences of the specified character from the original string. The function dynamically allocates memory for the modified string, copying the characters from the original string while skipping the specified character. It also null-terminates the modified string and frees the memory of the original string. The modified string is then returned. If there is an error in memory allocation, it exits the program with a failure message. This function ensures that memory is properly managed for the modified string.

- ## convertToLowercase Function:

this function will take a character string as a parameter and Iterates through each character in the string, converts each character to lowercase using the tolower function from the standard library.

- ## containsLetter Function:

  this function takes a character c and the random word string as a parameter , the purpose is to check if a specified character is present in a string.

- **Note**

All the last 3 functions will be used in the Guessing function.

- **Guessing letters Function:**

There are 5 variables in this function:

difficultyOption: Stores the user's choice of game difficulty.

correctGuessCounter: Counter for correct guesses.

missesCounter: Counter for incorrect guesses.

maxMisses: Maximum allowed incorrect guesses.

validInput: Variable to check the validity of user input.

First it includes a Set of the maximum misses based on difficulty, the default maximum number of misses is 6 but if the user want to change the option 2 must be chosen.

After that initialize the misses and correct Guess arrays with null characters. These arrays will be used to store incorrect and correct guesses, respectively.

The original word is stored in the originalWord variable using strdup. This is done to keep track of the original word for later display.

An array guessedLetters of size 26 is used to keep track of letters that have been guessed. This is done to prevent counting the same letter as a correct guess multiple times.

Guess a Letter:

The function enters a loop where the player makes guesses until the game ends.

The player is prompted to guess a letter.

The guessed letter is transformed to lowercase, and the removeLetter function is called to update the randomWord without the guessed letter.

If the guessed letter is present in the original word:

The function checks if the letter has already been guessed. If yes, a message is displayed, and the loop continues.

If the letter hasn't been guessed, it updates the correctGuess array and increments the correctGuessCounter.

It then displays a message indicating the correct guess and shows the positions of correct guesses in the word.

If the guessed letter is not in the original word, it updates the misses array and increments the missesCounter.

The loop continues until the maximum number of misses is reached or the player correctly guesses the entire word.

Later print the appropriate message based on whether the player has reached the maximum allowed misses (game over) or correctly guessed the entire word (victory)

Finally update the counters passed by reference to the calling function and return correct guess array.

Game Over and Victory Conditions: The game loop continues until either the maximum allowed misses are reached (maxMisses) or the player successfully guesses all the letters in the word (correctGuessCounter == wordLength). Depending on the outcome, a corresponding message is printed, including the original word if it's a game over situation.

Memory Management: The memory allocated for the originalWord is freed.

Finally, the function returns the correctGuess array.

Showgame() :

The showgame function is a simple utility function designed to visually represent the length of a word in a guessing game. It uses asterisks (*) to display the length of the word.

- ## PlayGame function :

The function first declares arrays to store correct guesses (correctGuess) and misses (misses) based on the chosen word length. Then Initialize counters for misses and correct guesses.

Calls the Guessing function to simulate the guessing process. Receives the result and updates the counters.

Prints the correct guesses if there are any; otherwise, informs the player that there were no correct guesses.

Prints the misses if there are any; otherwise, informs the player that there were no misses.

- ## getFirstMenuOption Function:

the function asks the user to choose between playing the game or exiting. Displays a menu with options to play (1) or exit (2).

Uses a do-while loop to validate user input, ensuring it is an integer and a valid menu option.

Clears the input buffer to remove non-integer input.

Returns the user's chosen option (1 or 2).

- ## getDifficultyOption Function:

the function asks the user to choose the difficulty level for the game.

Displays difficulty level options: Default (1) and Special (2).

Uses a do-while loop to validate user input, ensuring it is an integer and a valid difficulty option.

Clears the input buffer to remove non-integer input.

Returns the user's chosen difficulty level (1 or 2).

- ## launcher Function:

Acts as the main launcher for the game, allowing the user to play the game or exit.

Uses a do-while loop to repeatedly prompt the user for the main menu option (getFirstMenuOption).

If the user chooses to play (option 1), the playGame function is called with the game's linked list as an argument.

After playing, prints two newlines for better formatting.

Continues looping until the user chooses to exit (option 2).

Exits the program with a status code of 1.

- ## main () function:

the function first calls the openFile function to open a file named "words.txt" for reading.

Initializes a linked list (head is initially set to NULL).

Reads words from the file using fscanf and adds each word to the linked list using the addelement function.

Then closes the file, calls the launcher function with the linked list (head) as an argument.calls the freeList function to free the memory allocated for the linked list.