Hangman Documentation

Explanation of solution:

The main goal of the game: the user must guess the word that the program randomly selects.
Separate files are used to store words with the required number of letters ("words" folder). With scanf, the user enters a number between 3 and 7, which is the number of letters in a random word. After that, using the handleGetRandomWord function, open the file with the number entered by the user. Count the words in the file and create a 2D dynamic array with their number. After finding a random word in this array using the getRandomWord function.

Then the word hidden behind a dash is displayed. To do this, a dynamic array filled with dash characters is created. Variables are created to count correctly entered letters and mistakes made by the user. A "lives" variable is created and the default value is set to 10. Next, one will be subtracted from the "lives" value for each user mistake and display to the user how many attempts they have left to enter a letter and guess the word.

A “while loop” is used to test the user's letters until the number of mistakes is less than 10. If set to true, the following code is executed in the "while loop". If the value is 10, the game ends with the user losing.

Prompts the user for a letter using scanf. Check if the user has entered a letter. If not, then using the “while loop”, logic asks it for a new value until we get true (a letter, not another character).

A dynamic array is created, in which the letters already used by the user will be written. Subsequently, in the "while loop", after the user has entered his letter through the "for loop", it will be checked whether the entered letter is in this array. If so, prompt the user for a new letter. If not, continue with the next code.

Create a new variable “totalRight” with value 0.

Check if the letter entered by the user is in a hidden random word. Check it out with a "for loop". If true, show this letter instead of the dash (where it should be). Write plus one to "totalRight" and the variable we created to count correctly entered letters.

Then use an if loop to check the value of the “totalRight” variable. If it is equal to 1, then skip this loop and go to the next code. If it is equal to 0 (that is, if the

letter entered by the person is not in the word), then we take away one life and add one to the errors. Depending on the number of mistakes, logic outputs the corresponding section of the gallows drawing from the printGallows function.

At the end of the “while loop”, logic compares the number of letters in the word with the number of letters correctly entered by the user. If true, then the user has won. If false, then repeat the entire code from the “while loop” again until either the number of correctly entered letters equals the length of the word, or until the mistakes value becomes 10.
At the end of the program, the user is asked if he wants to repeat the game. If yes, then run the program again. If not, then we leave. For this we use a “while loop”.

List of the functions:

- checkUserLetters
- characterOrNot
- userNumber
- getRandomWord
- handleGetRandomWord
- lowerCase
- printGallows

void checkUserLetters(char *mainWord, int lengthOfWord)

input parameters:

char *mainWord	random word from file in lower case
int lengthOfWord	length of random word

returned result:

it receives any data from the calling function but it returns no values

description:

the main function that checks the letter entered by the user. The function receives the word to be guessed and its length

bool characterOrNot(char c)

input parameters:

char c	scanf value from user

returned result:

`true` if the value entered by the user is a letter

`false` if the value entered by the user is not a letter

description:

the function checks if the user has entered a character

char userNumber()

input parameters:

no input parameters

returned result:

letter like number between `three` and `seven` from user(char)

description:

the function asks the person for the number of letters in a word and returns its value

char *getRandomWord(char **array, int countWords)

input parameters:

char **array	dynamic array with words from the file
int countWords	the value of the number of words in the file

returned result:

`random word` from input array

description:

the function gets random word from array

char *handleGetRandomWord(char *nameOfFile**)**

input parameters:

char *nameOfFile	file name containing words with a certain number of letters

returned result:

random word from a file

description:

processing of obtaining a random word

char *lowerCase(char *nameOfFile**)**

input parameters:

char *nameOfFile	file name containing words with a certain number of letters

returned result:

returns a random word in lowercase

description:

the function converts letters to lowercase

void printGallows(int mistakes**)**

input parameters:

int mistakes	the count of incorrectly entered letters

returned result:

it receives any data from the calling function but it returns no values

description:

the function that prints the appropriate part of the gallows, depending on the number of errors made by the user