developer doc

solution explanation:

firstly, the program will print text to ask the user to choose a level (easy – hard).

If the value entered by user was not 2 or 1 then the program will keep asking the user to enter a valid value.

Then the program will ask if the value entered by user was 1 then the program will store the easy words text file as the file name and will set the maximum numbers of misses to 15. If not, the program will store the hard words text file as the file name and will set the maximum numbers of misses to 10.

```
if (Level_choice == 1)
{
    filename = "Easy_words_list.txt";
    NUM = 15;
}
else
{
    filename = "Hard_words_list.txt";
    NUM = 10;
}
```

Then the program will open the file, read the words, and store them in a string.

Then we will store the length of each word in the sting in an array.

And we will use a function to determine the number of the words in the list. (I will explain this function in detail later)

Then we will create a 2D array using dynamic memory to give each word the needed memory depends on its length.

```
char ** words_list = (char ** ) malloc(List_col * sizeof(char * ));
words_list[0] = (char * ) malloc(List_col * arr[0] * sizeof(char));

for (I = 1; I < List_col; ++I)
    words_list[I] = words_list[I - 1] + arr[I - 1];
```

Then we will open the file again and store the words in a string and then store them to the 2D array.

```
for (I = 0; fgets(input, 63, fp) != NULL ; I++)
{
    sscanf(input, "%s", words_list[I]);
}
```

Then the program will start counting the time.

Then the program will choose a random number to define a random word each time.

We will store the length of the word and will create another array (word_letters) with all elements are zero and each element will be one if the user guessed it right to know which character guessed right.

Then we will make while loop to know if the correct letters is the same as the number of word letters if not, we will continue in the function if yes, we will exit from the function.

Then we will print the word letters if the user guessed it right. If not, the program will print "-" we will use (word_letters) array to know that.

```
if (word_letters[N] == 1)
    printf("%c", words_list[random_number][N]);
else
    printf("-");
```

Then we will ask the user to enter some string and the program will exit from the while loop if the entered string was "give up", "exit" and "Quit". If not, the program will use the first letter as the guessed letter.

Then we will use (word_letters) array to check if the user entered a letter which he/she entered before if yes, we set letter_entereded_already to one.

```
if (word_letters[N] == 1)
{
    if (words_list[random_number][N] == gussed_letter)
    {
        letter_entereded_already = 1;
        break;
    }
```

If not, we will continue to see if the guessed letter in the word or not if yes, the program will increase the correct answer by 1.

```
if (gussed_letter == words_list[random_number][N])
{
    word_letters[N] = 1;
    Correct_letters++;
}
```

Then we will ask if the previous number of correct answer equal the correct number and it wasn't a guessed already letter that means it is a wrong guessed number then we will increase the misses number by one and ask if the misses number equal the maximum misses number if yes we exit from the loop.

If letter_entereded_already is one, then the program will notify the user that the letter is already entered.

The while loop will keep repeating until we exit, or the number of correct letters will be the same as word letters

After we get out of the loop the program will notify the user if he/she give up (if exit is set to one)

Or he/she lose (if wrong guesses were equal to maximum of misses)

If none then the user won.

```
if (exit == 1)
{
    printf("\nYou gave up! the word was: %s\n", words_list[random_number]);
}
else if (wrong_letters == NUM)
{
    printf("\nGame over, the word was: %s\n", words_list[random_number]);
}
else
    printf("\nCONGRATOLATIONS YOU WON!!\n");
```

And then the computer will ask the user if he want play again if yes then the program will set re_play to 1 if not the program will end

```
int K = 0;
while (K == 0)
{
    printf("\nDo you want to play again?\n 1- yes \n 2- No\n");
    printf("\nPlease enter a valid choice number: ");
    K = scan_a_number(K);
    K = choice_number(K);
    system("cls");
}
if (K == 1) Re_play = 1;
else
{
    Re_play = 0;
    printf("\nThank you for playing Hangman\n");
}
```

If the user did not enter a valid value, the program will keep asking until he/she enter a valid value.

Functions List:

```
int list_Words_number(char * filename)
{
    int i;
    char str[30];
    FILE * fp = fopen(filename, "r");
    for (i = 0; fgets(str, 30, fp) != NULL; i++) {}
    fclose(fp);
    return i;
}
```

this function will count how many words in a text file by receiving a pointer points to a string which is the file name.

then we will declare another string (char str [30]) to receive the word form the file after opening it.

After getting each word from the file to the string the counter will increase by one.

After there are no other words to get that means the list is over and will get out form the loop. And return the counter value.

```
int scan_a_number(int x)
{
    fflush(stdin);
    if (scanf("%d", & x) == 0)
        return 0;
    return x;
}
```

This function will make sure if the input was a number by getting the integer x as input and clear the stdin and then scan the number if it scanned then return it if not then return zero.

```
int choice_number(int x)
{
    if (x != 2 && x != 1)
        return 0;
    return x;
}
```

This function will receive an integer x and will return it if it is a one or two and will return zero if it is not.

```
int time_min(time_t s, time_t e)
{
    int T =(e - s);
    int min = T / 60;
    return min;
}
```

This function will receive the time start and the time end and will subtract the start from the end to get the time elapsed in second. After dividing it by 60 we will get the integer value of minuets. Then we return it.

```
int time_sec(time_t s, time_t e)
{
    int T = e - s;
    int sec = T % 60;
    return sec;
}
```

This function will receive the time start and the time end and will subtract the start from the end to get the time elapsed in second. Then we will get the reminder value which is the number of seconds. Then the program will return it.