Documentation of Hangman

General description

My program is Hangman. The main idea in this game is that, user has 5 attempts (it can be modified) and he/she tries to guess one letter in all attempts and by this way user tries to find the word which was chosen by the program randomly. In my program it is possible to play game in 5 different languages (English, Russian, Turkish, Azerbaijani and Hungarian). The length of words minimum is 3 and maximum is 14. There are 3 groups of word which are separated by their difficulties. Easy (3-7), medium(8-11), hard(12-14). Now I will explain all the parts of program clearly.

Functions

- **getDifficulty –** this function does not require any parameter. Return type of this function is "int". The return of this function will be used as length of word which is defined by user in this function and most important value for the following program and other functions. Variable "length" is the return of this function which is local for this. Variable "choice" is the int function which is scanned by user and use for determine difficulty of game.
- **checkLetter –** this is Boolean function, return type is "int". If function returns 1, it means condition is true, otherwise it is not. It has 2 parameters: "char* word", "char letter ". The purpose of this function is so simple, this one just check condition if any element of string "word" is equal to "letter" returns 1, but when the loop has finished and it has not returned yet, it means there is no character equal to "letter", and it returns 0.
- **findLetter –** this function's return type is void, clearly it returns nothing. Parameter "guess" is user's situation and this function exactly works on this string. So, variable "word" is string which defines word which is randomly selected and "letter" is the user's guess which he/she tries to know whether this word has this letter or not. Principle of this function is simple if we suppose "word" as array we can say that if any element of this array is equal to character "letter" we assign the same position of array (string pointer) "guess" to this letter.
- **checkGuess** – this function also works as Boolean logic. This is so important part of program, because in every attempts program has to check whether user found whole word or not. Clearly, if the word consists of 3 letters and user found all letters in his first 3 attempt. In this situation, user does not need other attempts. Shortly this function requires 2 inputs. Both of them are char pointer which refers to string. Principle is that checking all elements one by one, if there is inequality between characters of same position with 2 array of strings function returns 0. Till the end if function still not returns, it means all elements equal and it returns 1.

- **getLength** – this function is used for getting length of string. It requires pointer of char (string) and return its length. Of course return type is int. It works with this principle: number of elements = length of string and counter (it is variable "i") increases one in each step.
- **getLengthOFList** – it is almost same principle with **getLength** function. So, in here parameter is pointer which points to list structure. And one temporary pointer points to next in each step and in each step counter (i) increases one. Return type is "int" which determines length of list.
- **getRandomWord** - This function requires two parameters. First one is pointer which points to list which is listed with words with same length. Another input is char pointer which is defined in main function for getting random word, this function is void and does not return anything, it is just works on parameters. I used **"srand(time(0))"** function for getting random number.
  rand()%getLengthOFList(temp) – this means rand() function gives us one random integer, and getLengthOFList(temp) gives us number of elements which this list has. By this way, I get random number and remainder of this number when divided to length of list. It can be 0<n<length of list -1.
  And in the end, n-th element of list will be chosen and copied this string to address of "word".
- **chooseLanguage** – this is for extra of my program. In the main first this function should be called. It is for choosing language. Return type is pointer which points to file (FILE*), I assigned it file pointer in main function which is list of words in chosen language. This function does not require any parameters.

General Principle and Main Function

As I said first I used **chooseLanguage** function for finding list of words in chosen language and assigned its return to file pointer. Then I create list pointer which I defined by structure before. Firstly, "temp" is not pointing to something and "head" is null. After that I asked from user for choosing difficulty by using **getDifficulty** function. So, "while (fscanf (fp , "%s", c) != EOF)" with this loop I take all words from list and check if length is equal to defined length from user I take this word and add it to my list. The data of list structure is also pointer and for this I have to allocate memory for both list and data of list as well. And after this I have to add one line which is for pointer "next" which points next data in list, and copy this string this allocated memory. After filling the list with words with this length I used **getRandomWord** function for getting random word from this list. I added char array which is "guess" has same length with string "word". This array ("guess") is initialized with "#" this character. In the final, there is one more general while loop which is exactly heart of program. Generally this loop cannot be repeated more than "attempt" which has value "ATT" is defined before and it is fixed. The idea is that , in every guess of user the program has to be check if

this letter can be found in this word which is defined randomly, call this function **findLetter** for make known this character in this array which describes user's situation. And there is one more important point which I said calling **checkGuess** for checking user's guess if he/she found all letters the program does not need continue. In the end, if user's attempt finished but he/she could not find word I give last chance for guessing complete word. For this I added one more char array for using user's last guess and again I called **checkGuess** function for checking situation. Before finishing the program I have to close the file.