# Developer Documentation

## Goal of The Project

The goal of the project is to create a Hangman game. The project is created to be the most same as the real-life game for the use. The user will be able to play the game on the terminal according to the Original Rule. However, the user will be able to change the difficulty of the game, i.e., the user can change the number of times to miss. Moreover, the user can choose the length of the random word. Therefore, the user has almost complete control over the game.

## Solution

### Modules

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
```

The program uses “stdio.h”, “stdlib.h”, “time.h” and “string.h” modules. The “stdio.h” module is used to be able to run the most basics C program functions. The “stdlib.h” module is for creating dynamic arrays. The “time.h” module is used to for seed number for the random function. The “string.h” module is used to handle strings more easily.

### Data Structures

The program uses basics data types, static and dynamic arrays, and file data type.

```
char letter;
int length;
char str[20];


char *word = (char *)malloc((length + 1) * sizeof(char));
char **word_arr = (char**) malloc(sizeof(char*)*1);
word_arr[0] = (char *)malloc(sizeof(char)*20*1);

FILE *f;
```

### Algorithms

1. Print Out the Menu
    1. Option 1: Start the game
    2. Option 2: Show statistics of the guessed letters

3. Option 3: Exit

If the user choose Option 1,

1. Ask the user the length of the random word
2. Ask the user the difficulty level
3. Create a dynamic array with length the user inputted to store the random word
4. Create a 2D dynamic array for storing the words from a word file
5. Store the words from the file into the 2D array
6. Get the random word of the given length from the 2D array
7. Start the game –

   While the game is still operating

   a. Ask the user to guess a letter ( If it is an alphabet, do statistics of the letter)

   b. Check the letter whether it is in the correct word or not

   c. If it is not in the correct word increase the number of miss if it is in the word, reveal all the places of that number in the original word

   d. If the user has not guessed all the letters of the word or if he still has chances, return to stage **a.**

   e. If the user has guessed all the letters or he has no more chances, the game will end, Save the statistics to a file.

If the user choose option 2,

1. The program will show the statistics of the guessed letters from the file in where the statistics is saved.

If the user choose option 3,

1. exit the program

## Lists of Functions

| **Syntax** | **Parameter** | **Description** | **Return Value** |
|---|---|---|---|
| `void title()` | No parameter | prints “HANGMAN” with Ascii Art | no return value |
| `int random_num(int max_words)` | `int max_words` = maximum number of words | generate random number between 0 and maximum number of words | returns a random number of an integer type |

| Syntax | Parameter | Description | Return Value |
|---|---|---|---|
| `char **create_words(char filename[], char **words,int *max_words)` | 1. `char filename[]` = the name of the file which stores words list 2. `char **words` = a 2D array to store the words, 3.`int *max_words` = address of a variable to store maximum number of words | store words from the file to the 2D array and return that array and return the maximum number of words from the parameter called *max_words* | return a 2D array which stores word from the file |
| `void randomWord(int length, char **words, char *w,int max_words)` | 1. `int length` - length of the random word, 2. `char **words` - 2D array that stores words, 3. `char *w`- an array to store the random word, 4.`int max_words` = maximum number of words | from the word_array get a random word and pass it to the array *w* | no return value |
| `void print_dash(int length)` | `int length` = length of the correct word | print dashes according to the length variable | no return value |
| `void print_arr(char *guess)` | `char *guess` = the array to be printed | print the element of the array “guess” | no return value |
| `int already_guess(char letter, char *guess)` | 1. `char letter` = the letter to be checked, 2. `char *guess` = the array of correctly guessed letter | To check whether a letter is already guessed or not . If the letter is correct and has been already guessed, the letter will be checked as guessed | return – 1 if the letter has already been guessed, return - 0 if the letter has not been guessed |

| Syntax | Parameter | Description | Return Value |
|---|---|---|---|
| `int not_in(char letter, char *word)` | 1. `char letter` = the letter to be checked, 2. `char *word` = the array of original word (correct word) | To check whether a letter is in the original word if the letter is not in the original word the letter will be checked as not in the original word | return – 1 if the letter is not in the original word, return - 0 if the letter is in the original word |
| `void Words_statistics(char letter, int *num_words)` | 1. `char letter` - the letter to be counted, 2. `int *num_words` - the array which stores the previous statistics data for each letter | This function will do the statistics of the letters guessed by user. This will count the number of the letters user guessed and add it to the previous data contained in the array | no return value |
| `void get_statistics(char filename[], int *num_words)` | 1. `char filename[]` - name of the file which store the previous statistics data, 2. `int *num_word` - the array to store previous data | Get data from the file and store it in the given array | no retrun value |
| `void read_statistics(char filename[])` | `char filename[]` - the name of the file which store the previous statistics data | Read and prints out the previous data | no return value |
| `void write_file(int *num_words)` | `int *num_words` - array which stores the statistics data | write the data from the array into a file called **words_statistic.txt** | no return value |
| `void hangman()` | no parameter | display hangman game over picture | no return value |
| `int reduce_chance(int miss,char *guess,int chance)` | 1. `int miss` - number of times the user has missed, 2. `char *guess` - the array of correctly guessed letter 3. `int chance` - the number of chance the user has | print guessed words, increase miss count, print remaining chances | no return value |

| Syntax | Parameter | Description | Return Value |
|---|---|---|---|
| `void guess_word(char *word, int chance, char filename[])` | 1. `char *word` = the random word generated to be guessed, 2. `int chance` - the number of the times the user can miss, 3. `char filename[]` - name of the file which store the previous statistics data | This is the source of the game. This function will receive a word and processes it according to the rule of the Hangman game This will check whether a word is guessed or in the correct word. Decide whether the user wins or lose the game | no return value |
| `void play_game()` | no parameter | This is Hangman_game.exe function. In this function all the necessary function to run the game are called. Dynamics arrays to store words, data and random word are created. Asked the user to input length of the random word, number of chances. | no return value |
| `void print_menu()` | no parameter | This function prints the menu and ask the user to type one of the option. This function is also a recursive function. Whenever user chooses an option after carrying it out, it will return to the print function. Exit upon choosing exit option. | no return Value |

## Testing

When the menu is printed out,

> 1. choosing option 1

```
To play game press 1
To see the statistics of letters press 2
To exit press 3
1
Type the length for a random word (min-3,max-16)
5
Choose the number of chances(Difficulty : 1 = The Hardest , 10 = the easiest)
5
_ _ _ _ _           <Chances: 5>
Guess a letter
e
The letter is not in the word!
_ _ _ _ _           <Chances: 4>
i
The letter is not in the word!
_ _ _ _ _           <Chances: 3>
a
_ a _ _ _           <Chances: 3>
o
_ a _ o _           <Chances: 3>
r
_ a _ o r           <Chances: 3>
b
The letter is not in the word!
_ a _ o r           <Chances: 2>
t
The letter is not in the word!
_ a _ o r           <Chances: 1>
c
The letter is not in the word!
_ a _ o r           <Chances: 0>
No Chances left!!!
 +---+
 |   |
 O   |
/|\  |
/ \  |
     |
=========
You couldn't guess all the letters! You lose!The word is [ favor ]

To play game press 1
To see the statistics of letters press 2
To exit press 3
```

2. choosing option 2

```
To play game press 1
To see the statistics of letters press 2
To exit press 3
2
Printing the number of letters guessed
[a]: 42  [b]: 15  [c]: 16  [d]:  4  [e]: 30  [f]:  9  [g]: 12  [h]: 12  [i]: 22  [j]:  0  [k]:  2  [l]:  2  [m]:  7
[n]: 10  [o]: 10  [p]:  1  [q]:  3  [r]: 16  [s]: 11  [t]: 14  [u]:  7  [v]:  5  [w]:  5  [x]:  0  [y]:  1  [z]:  1

To play game press 1
To see the statistics of letters press 2
To exit press 3
```

3. choosing option 3

```
To play game press 1
To see the statistics of letters press 2
To exit press 3
3
PS C:\Users\Legion 5\Desktop\BOPhomework>
```

> 4. typing other than 1,2,3

```
|__|  |__| /__/     \__\ |__| \__|  \______| |__
To play game press 1
To see the statistics of letters press 2
To exit press 3
5
Please choose only from the given options
To play game press 1
To see the statistics of letters press 2
To exit press 3
abc
Please choose only from the given options
To play game press 1
To see the statistics of letters press 2
To exit press 3
```

## During the Game

> 1. Choosing between 3 and 16 for length and choosing between 1 and 10 for difficulty

```
Type the length for a random word (min-3,max-16)
7
Choose the number of chances(Difficulty : 1 = The Hardest , 10 = the easiest)
5
_ _ _ _ _ _ _          <Chances: 5>
Guess a letter
```

> 2. typing only one letter at each time

```
a
_ _ _ _ _ a _          <Chances: 5>
e
_ e _ _ _ a _          <Chances: 5>
i
The letter is not in the word!
_ e _ _ _ a _          <Chances: 4>
o
The letter is not in the word!
_ e _ _ _ a _          <Chances: 3>
c
The letter is not in the word!
_ e _ _ _ a _          <Chances: 2>
```

3. Choosing other that between 3 and 16 for length and other than between 1 and 10 for difficulty

```
Type the length for a random word (min-3,max-16)
20
The maximum length can only be 16. Therefore it is set to 16
Choose the number of chances(Difficulty : 1 = The Hardest , 10 = the easiest)
20
The maximum number of chance can only be 10. Therefore it is set to 10
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _        <Chances: 10>
Guess a letter
```

4. typing not letter for gussing

```
_ _ _ _ _ _ _          <Chances: 7>
Guess a letter
.
Please type letter only
_ _ _ _ _ _            <Chances: 6>
%
Please type letter only
_ _ _ _ _ _            <Chances: 5>
2
Please type letter only
_ _ _ _ _ _            <Chances: 4>
abc
Type only one letter at each time
_ _ _ _ _ _            <Chances: 3>
```

Winning the Game

```
e n t h u s i a s t i c a l l y       <Chances: 2>
You win!
You have guessed all the letter

To play game press 1
To see the statistics of letters press 2
To exit press 3
```

### Losing the game

```
Guess a letter
g
The letter is not in the word!
_ _ _ _ _          <Chances: 0>
No Chances left!!!
 +---+
 |   |
 O   |
/|\  |
/ \  |
     |
=========
You couldn't guess all the letters! You lose!The word is [ minor ]

To play game press 1
To see the statistics of letters press 2
To exit press 3
```

*This is the end of test cases*

---