Group “Apes Together Strong”

Habib Chandio: CT-24080

Ali-Jan: CT-24092

Hamadullah: CT-24097

Abdul-Bari: CT-24086

# Project Report: *Hangman* Game in C (Terminal)

Project Members:

| | | |
|---|---|---|
| 1. | Habib Chandio – Project Leader and Developer | Roll Number: CT-24080 |
| 2. | Ali-Jan – Logic Designer and Tester | Roll Number: CT-24092 |
| 3. | Hammadullah-Lakho – Game art Designer | Roll Number: CT-24097 |
| 4. | Abdul Bari – UI Developer | Roll Number: CT-24086 |

This project is based on the Hangman game, where you have to guess a word with a limited number of lives. This game has 10 levels, 6 lives for each level, and three difficulties: Easy, Medium, and Hard. The game also has a leaderboard. The word selection and leaderboard are managed through file handling. This game utilized loops and nested loops for game logic. It also features visual representation.

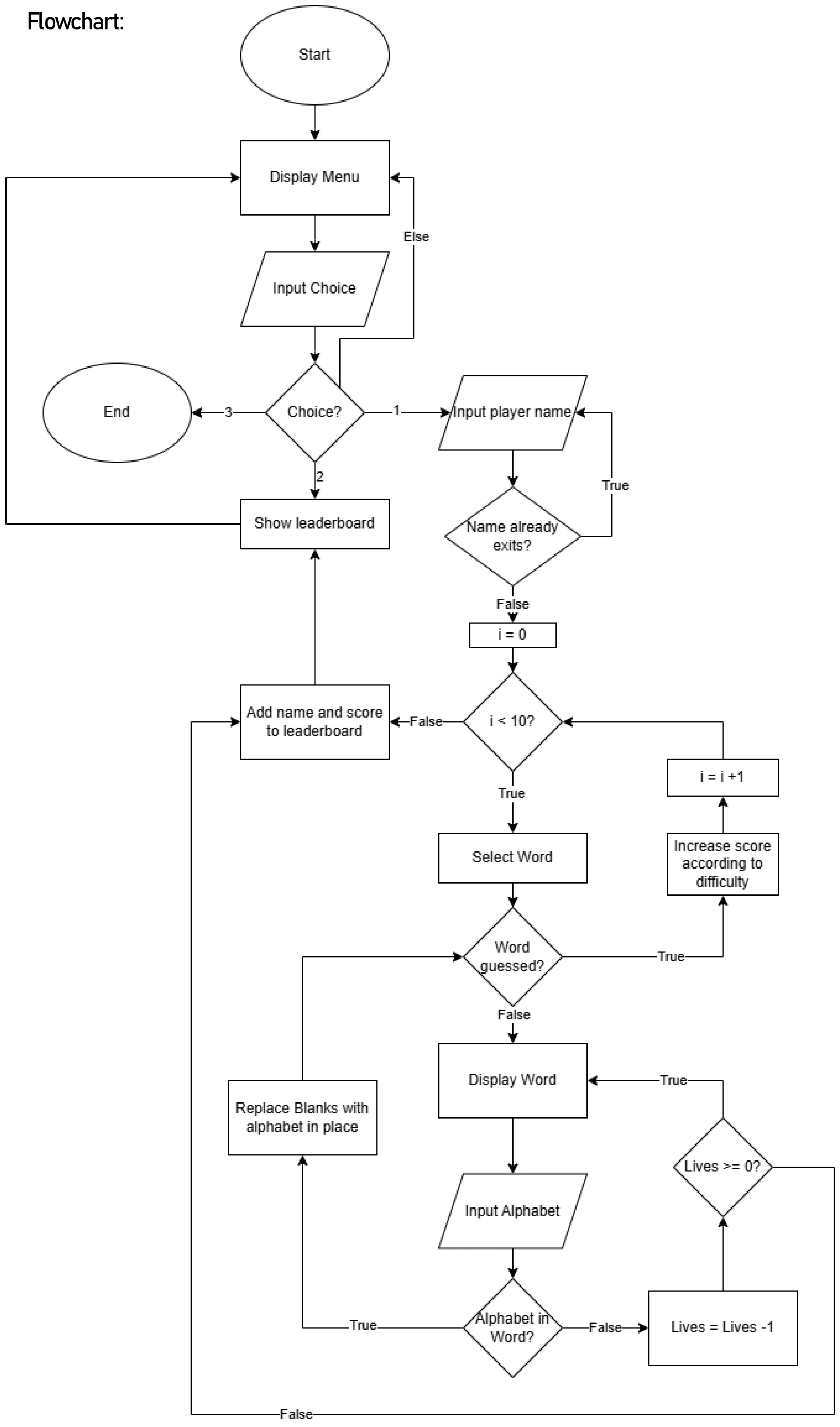
Project Functionalities:

1. Word Guessing Game:
   Players guess the letters of a hidden word within 6 attempts per word.
2. Dynamic Feedback:
   The program visually displays the guessed letters and the progress of the "hangman."
3. Scoring System:
   Players score points based on the number of correct guesses and word difficulty.
4. Difficulty Levels:
   Game difficulty increases as players progress through the game increasing word complexity.
5. Leaderboard:
   Top scoring player username displayed.

Game Overview

- First Select from the menu
- 1 to Start game
- 2 to See Leaderboard
- 3 to quit
- A player guesses the letters of a hidden word.
- They have a total of 6 incorrect guesses per word.
- Correct guesses reveal the letters in the word, while incorrect guesses reduce remaining attempts.

Flowchart:



Start
Display Menu
Input Choice
Choice?
Else
3
End
1
Input player name
2
Show leaderboard
Name already exits?
True
False
i = 0
i < 10?
False
Add name and score to leaderboard
True
Select Word
i = i +1
Increase score according to difficulty
Word guessed?
True
False
Display Word
True
Replace Blanks with alphabet in place
Lives >= 0?
Input Alphabet
Alphabet in Word?
True
False
Lives = Lives -1
False

Datatype

1. int: lives, min, max, flag, choice, score, i, j, k, lineNo, tempInt, currentLine
2. char: name, alpha, guessWord, word, line, copy, tempChar
3. char[ ][ ]: names
4. int[ ]: scores
5. FILE*: file

---

Function Purposes and Descriptions

---

1. void viewLeaderboard()

Purpose:
Displays the top 10 players from the leaderboard file.

Details:
Reads the leaderboard.txt file line by line and prints each line to the terminal. Stops after displaying 10 entries or if the file ends. Closes the file after use.

Input: Nothing

Output: Shows leaderboard(Returns nothing)

---

2. int checkName(char* name)

Purpose:
Checks if a player's name exists in the leaderboard.

Details:
Reads leaderboard.txt line by line, compares each name in the file with the input name using strcmp. Returns 1 if the name exists, otherwise 0. Ensures no newline characters in the comparison.

Input: String

Output: Integer

---

3. void addToLeaderboard(char* name, int score)

Purpose:
Adds a new player's name and score to the leaderboard.

Details:
Appends the player's name and score to leaderboard.txt in a formatted manner. After writing, it calls sortLeaderboard() to update and reorder the leaderboard.

Input: String and Integer

Output:  Adds to leaderboard(Returns Nothing)

---

4. void sortLeaderboard()

Purpose:
Sorts the leaderboard entries in descending order of scores.

Details:
Reads all entries into arrays for names and scores, sorts them using bubble sort, and writes back the sorted entries to the file. Ensures the leaderboard reflects the highest scores first.

Input: Nothing

Output: Sorts leaderboard(Returns nothing)

---

5. void getWord(char* word, int line)

Purpose:
Retrieves a specific word from the words.txt file based on the line number.

Details:
Iterates through the lines of words.txt until it matches the specified line. If the line number is invalid, an error is printed.

Input: String and Integer

Output: modifies input string(Returns nothing)

---

6. int randInteger(int max, int min)

Purpose:
Generates a random integer within a given range.

Details:
Uses the rand() function to return a random number between the specified min and max values. Ideal for randomizing game elements like word selection.

Input: Integers(min, max)

Output: Integer

---

7. void printHangman(int lives)

Purpose:
Displays the hangman ASCII art based on remaining lives.

Details:
Prints different stages of the hangman, from an empty scaffold (6 lives) to a completed hangman (0 lives). Provides visual feedback on player progress.

Input: Integer

Output: Shows current state of hangman(Returns Nothing)

---

8. void getHint(char* currentWord, char* guessWord)

Purpose:
Generates a hint by revealing some letters of the current word.

Details:
Randomly selects and reveals approximately one-third of the letters in the currentWord. Updates the guessWord to reflect the revealed letters and ensures duplicates of revealed letters are also shown.

Input: Two Strings

Output: Modifies 'guessWord' string (Returns nothing)

---

Source Code

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

void viewLeaderboard();
int checkName(char* name);
void addToLeaderboard(char* name, int score);
void sortLeaderboard();
void getWord(char* word, int line);
int randInteger(int max, int min);
void printHangman(int lives);
void getHint(char* currentWord, char* guessWord);

int main()
{
      int lives = 6, min, max, flag = 0, choice = 0, score = 0;
    char name[20], alpha, guessWord[50], word[20];
      srand(time(NULL));
      do{
            printf("-------------WELCOME TO HANGMAN-------------\n\n");
            printf("1. Start\n2. Leaderboard\n3. Quit\n");
            printf("Selection(1 - 3): ");
            scanf(" %d", &choice);
            system("cls");
```

```
if(choice == 1){
    do{
        printf("Enter your name: ");
        scanf(" %s", &name);
        if(checkName(name) == 1){
            printf("Name already exists.\n");
        }
        else
            break;
    }while(1);
    system("cls");
    for(int i = 0; i < 10 && lives > 0; i++){
        if(i < 2){
            min = 1;
            max = 30;
        }
        else if(i < 5){
            min = 30;
            max = 60;
    }
    else{
            min = 60;
            max = 100;
    }
    getWord(word,randInteger(max, min));
        for (int j = 0; j < strlen(word) - 1; j++){
            guessWord[j] = '_';
        }
```

```
                  guessWord[strlen(word) - 1] = '\0';
            word[strcspn(word, "\n")] = '\0';
            guessWord[strcspn(guessWord, "\n")] = '\0';
            getHint(word,guessWord);
                  while(lives>0) {
                        flag = 0;
                        if(i < 2)
                          printf("DIFFICULTY: EASY\n");
                      else if(i < 5)
                          printf("DIFFICULTY: MEDIUM\n");
                      else
                          printf("DIFFICULTY: HARD\n");
                      printHangman(lives);
                        printf("\n%s\n\n",guessWord);
                        printf("Guess a character: ");
                  scanf(" %c", &alpha);
                  alpha = toupper(alpha);

                  for (int k = 0; k < strlen(guessWord); k++){
                          if (alpha == word[k]){
                              flag = 1;
                              guessWord[k] = alpha;
                      }
                  }
                        system("cls");
                        if (flag == 0){
                              lives--;
                      }
                        if (lives <= 0){
                              printf("You gussed wrong!!!\n");
                          break;
                      }
                      if(strcmp(guessWord,word) == 0){
                        printf("You guessed correct!!!\n");
                              if(i < 2)
                              score += 5;
                          else if(i < 5)
                              score += 10;
                          else
                              score += 15;
                          lives = 6;
                          break;
                        }
                  }
                  printf("The word was: %s!\n", word);
                  Sleep(1000);
                  system("cls");
            }
            printf("You scored: %d!!!!\n", score);
            addToLeaderboard(name, score);
            viewLeaderboard();
            printf("\n");
            lives = 6;
            score = 0;
      }
      else if(choice == 2){
```

```c
            viewLeaderboard();
            printf("\n");
        }
        else if(choice == 3){
            printf("Thanks for playing.\n");
            break;
        }
        else
            printf("Invalid input\n");
     }while(1);
    return 0;
}

void viewLeaderboard(){
    char line[100];
    int count = 1;
    FILE* file = fopen("Files/leaderboard.txt", "r");
    printf("--------Leaderboard--------\n\n");
    printf("%-20s %5s\n\n", "NAMES", "SCORE");
    while(fgets(line, sizeof(line), file)){
        printf("%s", line);
        count++;
        if(count == 11){
            fclose(file);
            return;
        }
    }
    fclose(file);
```

```c
}

int checkName(char* name){
      char line[100], copy[50];
      FILE* file = fopen("Files/leaderboard.txt", "r");
      while(fgets(line, sizeof(line), file)){
            sscanf(line, "%49s", copy);
            name[strcspn(name, "\n")] = 0;
            if(strcmp(name, copy) == 0){
                  return 1;
            }
      }
      return 0;
}

void addToLeaderboard(char* name, int score){
      FILE* file = fopen("Files/leaderboard.txt", "a");
      name[strcspn(name, "\n")] = 0;
      fprintf(file, "%-20s %5d\n", name, score);
      fclose(file);
      sortLeaderboard();
}

void sortLeaderboard(){
      char names[100][50], line[100], tempChar[50];
      int scores[100], lineNo = 0, tempInt = 0, index = 0;
      FILE* file = fopen("Files/leaderboard.txt", "r");
      while(fgets(line, sizeof(line), file)){
        sscanf(line, "%s %d", names[lineNo], &scores[lineNo]);
        lineNo++;
    }
    fclose(file);
    for(int i = 0; i < lineNo; i++){
        for(int j = i + 1; j < lineNo; j++){
            if(scores[i] < scores[j]){
                tempInt = scores[i];
                scores[i] = scores[j];
                scores[j] = tempInt;
                strcpy(tempChar, names[i]);
                strcpy(names[i], names[j]);
                strcpy(names[j], tempChar);
            }
        }
    }
    file = fopen("Files/leaderboard.txt", "w");
    while(index < lineNo){
        fprintf(file, "%-20s %5d\n", names[index], scores[index]);
        index++;
    }
    fclose(file);
}

void getWord(char* word,int line){
    int currentLine = 1;
    FILE* file = fopen("Files/words.txt", "r");
    while(fgets(word, 20, file)){
        if(currentLine == line){
            fclose(file);
        return;
        }
        currentLine++;
    }
    printf("getWord(): Incorrect Index.\n");
    return;
}

int randInteger(int max, int min){
    return (rand() % (max - min + 1)) + min;
}

void printHangman(int lives){
    switch(lives) {
    case 6:
        printf("  +---+\n"
               "  |   |\n"
               "      |\n"
               "      |\n"
               "      |\n"
               "      |\n"
               "=========\n");
        break;
    case 5:
        printf("  +---+\n"
               "  |   |\n"
               "  O   |\n"
               "      |\n"
               "      |\n"
               "      |\n"
               "=========\n");
        break;
    case 4:
        printf("  +---+\n"
               "  |   |\n"
               "  O   |\n"
               "  |   |\n"
               "      |\n"
               "      |\n"
               "=========\n");
        break;
    case 3:
        printf("  +---+\n"
               "  |   |\n"
               "  O   |\n"
               " /|   |\n"
               "      |\n"
               "      |\n"
               "=========\n");
        break;
    case 2:
        printf("  +---+\n"
               "  |   |\n"
               "  O   |\n"
```

```c
                   " /|\\  |\n"
                   "      |\n"
                   "      |\n"
                   "=========\n");
            break;
        case 1:
            printf("  +---+\n"
                   "  |   |\n"
                   "  O   |\n"
                   " /|\\  |\n"
                   " /    |\n"
                   "      |\n"
                   "=========\n");
            break;
        case 0:
            printf("  +---+\n"
                   "  |   |\n"
                   "  O   |\n"
                   " /|\\  |\n"
                   " / \\  |\n"
                   "      |\n"
                   "=========\n");
            break;
        default:
            printf("Invalid number of lives!\n");
            break;
    }
}
```

```

void getHint(char* currentWord, char* guessWord) {
      int j=0;
      do{
      int i = rand() % strlen(currentWord);
      guessWord[i] = currentWord[i];
      j++;
      for (int k = 0; k < strlen(guessWord); k++){
                  if (currentWord[k] == currentWord[i]){
                        guessWord[k]=currentWord[i];
                  }
            }
      }while ((strlen(currentWord)/3)>j);
    return;
}
```

Members Contribution

| Members | Contribution |
|---|---|
| Habib Chandio | Developed file handling and scoring system, debugged errors, optimized game performance and word selection. |
| Ali-Jan | Designed the main game flow, debugged errors, and logic functions. |
| Hammadullah Lakho | ASCII art for the Hangman. |
| Abdul Bari | Designed the UI/UX for the terminal |