Individual Software Game Project - Wordle

B37VB – Praxis Programming - 2024-2025

Professor: Girish Balasubramanian

Fatima Abbas (H00471066)

April 14, 2025

Introduction

This report documents the design, development and functionality of a simplified version of the popular word guessing game Wordle, implemented in C language. The objective of the game is for the player to guess a randomly selected word of a specified length within a limited number of attempts. This project aims to reinforce programming concepts such as loops, conditionals, functions, user input validation, and string manipulation, arrays and the use of standard libraries.

Game description & User Instructions

The game begins by displaying a welcome message and prompting the player to choose a desired word length (4, 5, or 6 letters).

"1" = 4 letters

"2" = 5 letters

"3" = 6 letters

According to what the user selected; a random word is chosen from a predefined list. If an invalid input is given, the game defaults to a 5-letter word and tells the user. The player is then prompted to enter guesses for the hidden word. Each guess is validated and evaluated. Color-coded feedback is provided after each guess indicating how close the player is to the actual word.

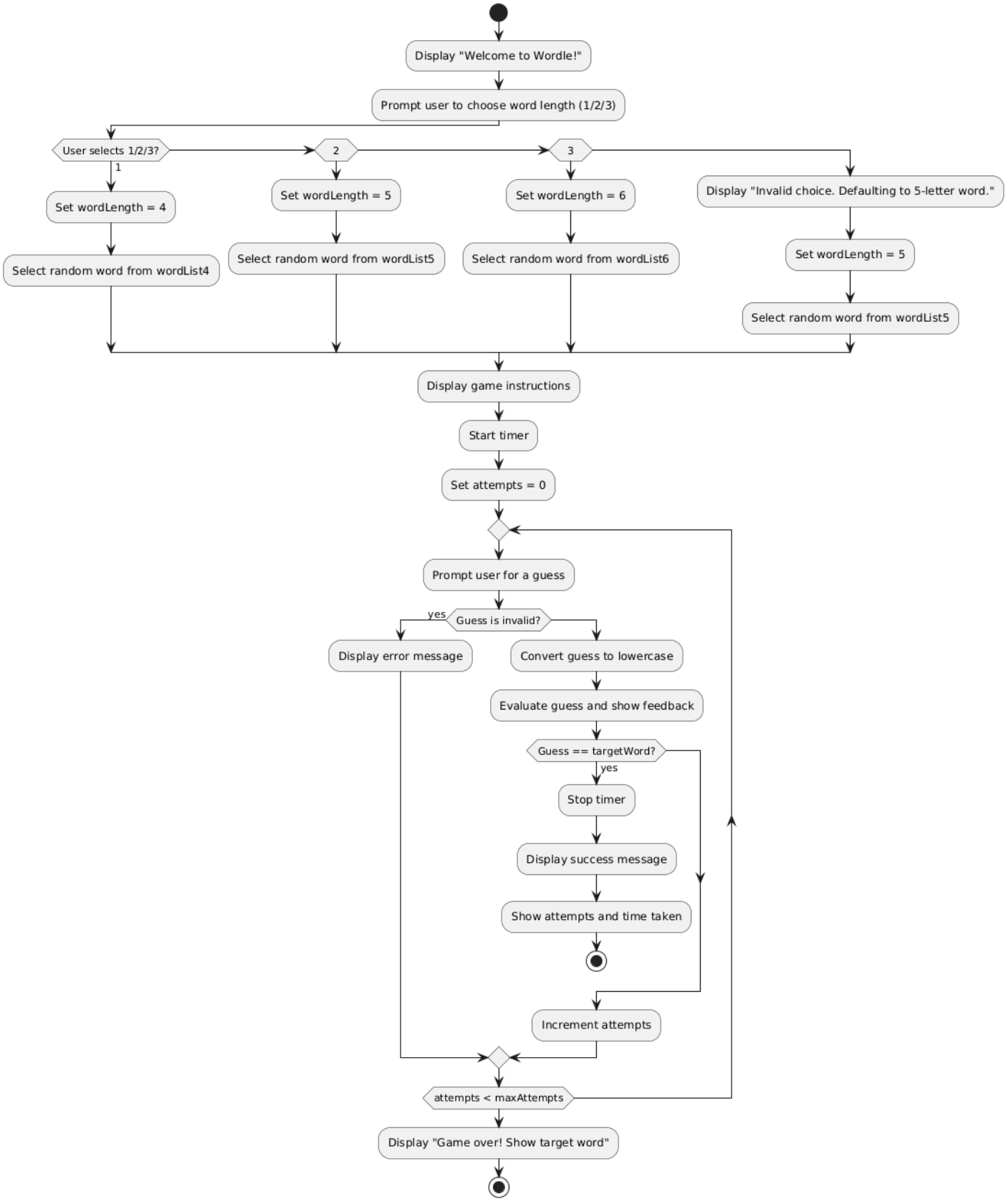
Green: Correct letter and correct position

Yellow: Correct letter but wrong position

Red: Letter not in the word

The player has a limited number of attempts to guess the word. If they guess it correctly within the allowed attempts, a success message and their time taken is displayed. Otherwise, a game-over message reveals the correct word.

Flowchart diagram



Code Structure and Functionality

1. Word selection:
   A random word is chosen using *rand()* based on the user's chosen length from the respective predefined word list (*wordList4, wordList5, wordList6*).

2. Input handling:
   The program ensures the entered guess has the correct length and that all characters are alphabetic (using *isalpha*). It handles invalid inputs by prompting again or using default values. The program also converts the input to lowercase using *tolower* to standardize the guesses and make the comparison easier.

3. ANSI colour output:
   To simulate the feedback just like the original wordle, ANSI escape codes are used.
   - *#define GREEN "\x1B[32m*" for green
   - *#define YELLOW "\x1B[33m"* for yellow
   - *#define RED "\x1B[31m"* for red
   - *#define RESET "\x1B[0m"* to reset the terminal colour after each letter

4. Guess evaluation (*evaluateGuess*):

   In the first pass, the program identifies and marks the correct letters in correct positions(green). In the second pass, it searches for correct letters in wrong positions(yellow). The letters that are not present in the word are marked red. *usedTarget[]* prevents the program from overcounting letters that have already been matched.

5. Timing:
   The game tracks the time taken from the start of the guessing phase until a correct guess is made. The stopwatch is implemented with *clock_t = clock()* and the time is calculated using *CLOCKS_PER_SEC*.

Challenges and Debugging

Some of the challenges faced during the development of this game include:

- Handling different cases for invalid input.
- Designing a coloured feedback mechanism to help guide the player.
- Ensuring the timer worked accurately

The program was debugged by carrying out various testing scenarios to ensure the game is efficient.

Conclusion

All in all, the wordle game was successful. Concepts such as input validation, randomization, string manipulation and time tracking were reinforced throughout this project. The project demonstrated the flexibility of C programming for interactive terminal applications. The game could also be further developed in both functionality and user experience by adding a graphical interface to add visuals to the game.

Video demonstrating the game

https://youtu.be/fFbaV7QpiFo