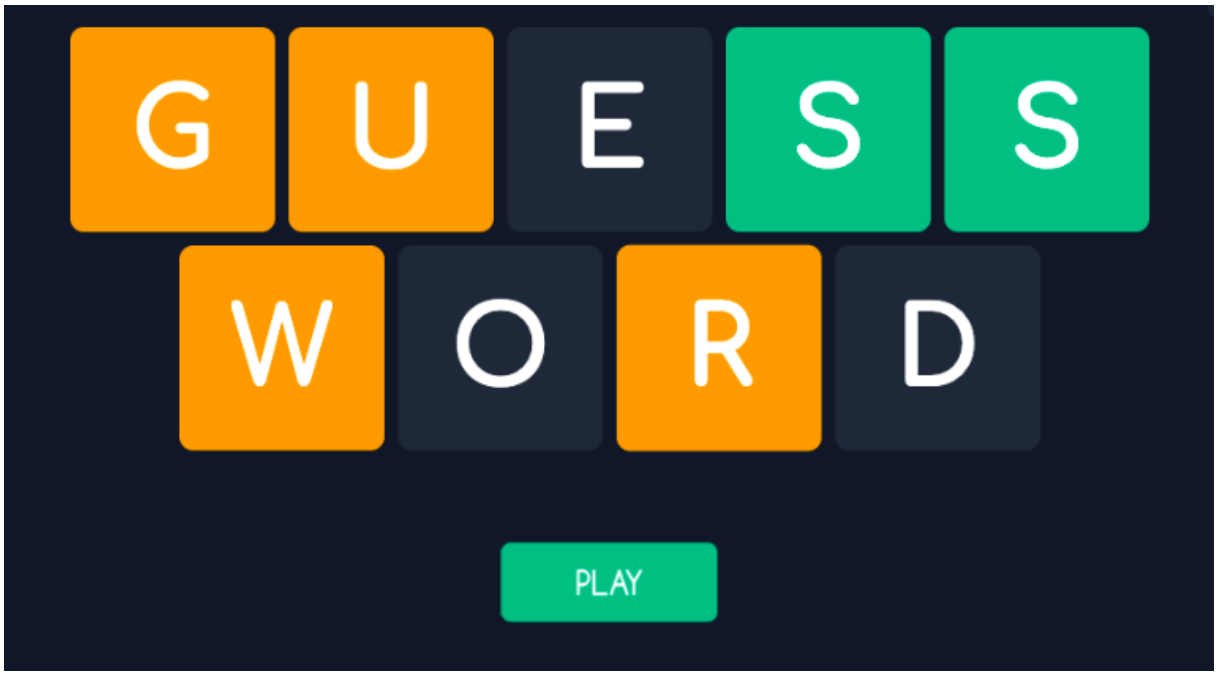
Basics Of Programming

Developer Documentation

Guess The Word Game



Contents

List of Figures

I. Overview

This developer documentation describes the code for a text-based word guessing game called ”Guess The Name.” The game allows the player to guess characters in a hidden word, and it provides features like choosing the word length and tracking the player's score based on the number of misses and time taken. This document will provide an overview of the solution, modules, data structures, algorithms, and a list of functions with explanations.

II. Solution Overview

The ”Guess The Name” game is implemented in C and consists of the following key modules and features:

A. File I/O

- The game reads a list of words from a text file (”names.txt”) using the `readFile` function and stores them in a single linked list.
- The `writeFile` function allows users to add words to the text file for later gameplay.

B. Gameplay

- The core gameplay logic is in the `GauseName` function, where players guess characters in a word, one by one.
- Players are allowed a certain number of misses based on the selected difficulty level.
- Players can choose to play with a randomly selected word or select a word of a specific length.

C. Scoreboard

- The game tracks and displays player statistics, including the number of misses, time taken, and overall score.

D. User Interface

- The game includes a welcome page and menu for players to choose whether to add words or start playing.
- Players can choose the difficulty level of the game.

III. Data Structures

The primary data structures used in the game include:

- `Node`: Is used for creating a singly linked list. The structure contains two members:
- `char *data`: This member is a pointer to a character array (string) and is intended to store data associated with each node in the linked list. It represents the information that you want to store in the list.
- `struct Node *next`: This member is a pointer to another structure of the same type, specifically a pointer to the next node in the linked list. It is used to establish the connection between nodes in a singly linked list, allowing you to traverse the list sequentially by following these pointers.
- `char** WordArray`: A 2D dynamic array used to store the list of words read from the text file.
- `char* TempArray`: A dynamic array used to store the current state of the word being guessed.

IV. Algorithms

The code employs various algorithms for game logic, including:

- Random word selection for gameplay.
- Character guessing and tracking.
- Calculating player scores based on misses and time taken.

V. List of Functions

Below is a list of functions used in the game, along with their input parameters, return values, and brief explanations:

1) `Node *CreateNode(char *data)`
   - This function is responsible for creating a new node and initializing it with the provided data.
   - It takes 'data' as a parameter, which is the information to be stored in the new node.
   - It returns a pointer to the newly created node.

2) `Node * insertNode(Node *head, char *data)`
   - This function is used to insert data into a node and add it to the linked list.
   - It requires two parameters: 'head' (a pointer to the head of the linked list) and 'data' (the data to be inserted into the node).
   - It returns the updated head of the linked list.

3) `Node *ReadtextFile(Node *head, int *totalWord)`
   - This function reads data from a text file and populates the linked list.
   - It takes two parameters: 'head' (a pointer to the head of the linked list) and 'totalWord' (a pointer to an integer that stores the total number of words read).
   - It returns the head of the linked list after it has been populated.

4) `void dispose(Node *head)`
   - This This function is used to free the memory utilized by the linked list.
   - It takes 'head' as a parameter, which is a pointer to the head of the linked list.

5) `char** StorageOflist(Node *head, int lines)`
   - This This function is responsible for printing the linked list and returning it as a 2D array of strings.
   - It requires two parameters: 'head' (a pointer to the head of the linked list) and 'lines' (the number of lines in the linked list).
   - It returns a 2D array representing the linked list.

6) `void writeFile()`
   - Allows users to add words to the "names.txt" file for gameplay.
   - Reads words from the user and appends them to the file.

7) `int WelcomePage()`
   - Returns 1 if the user chooses to add words or 0 if they want to start playing.
   - Displays a welcome message and menu to the user.

8) `unsigned int randomizer(int max)`
   - Input Parameters: `int max` (the upper limit for random number generation).
   - Returns a random number from 0 to `max`.
   - Used for various random selections in the game.

9) `char* wordSelection(int length, char** array)`
   - Input Parameters: `int length` (the length of the array), `char** array` (the array of words).
   - Returns a dynamically allocated string representing a randomly selected word from the array.
   - The function also ensures the word is copied correctly.

10) `int Difficulty()`
   - Allows the user to select the difficulty level (number of allowed misses).
   - Returns the selected difficulty level as an integer.

11) `void ScoreBoard(int missedC, int wordsize, double time, int win)`
   - Input Parameters: `int missedC` (number of missed characters), `int wordsize` (word length), `double time` (time taken), `int win` (1 for a win, 0 for a loss).
   - Prints a scoreboard displaying game results, including time, misses, and score.

12) `int GauseName(char* word)`
   - Input Parameters: `char* word` (the word to be guessed).
   - Returns 1 for a win and 0 for a loss.
   - Implements the core game logic, including character guessing and tracking.

13) `char* wordSearchOfLength(int lengthOfWord, int length, char** array)`
   - Input Parameters: `int lengthOfWord` (desired word length), `int length` (length of the word array), `char** array` (the array of words).
   - Returns a dynamically allocated string representing a word of the specified length.
   - Used to select words of a specific length for gameplay.

14) `int main()`
   - The main function orchestrates the game and provides the user interface.
   - Allows the player to choose word length and replay the game.

VI. Summary

The ”Guess The Name” game is a text-based word guessing game that allows users to read words from a file, add their own words, choose difficulty levels, and play the game. The code employs dynamic memory allocation, file I/O, and various game logic elements to provide an interactive gaming experience.