Introduction

This document provides an overview of the Hangman game implemented in C++. Hangman is a classic word-guessing game where the player attempts to guess the letters of a hidden word. Each incorrect guess reduces the number of lives, represented by the drawing of a hangman. The game ends when the player either guesses the entire word or runs out of lives.

Code Structure

The program is structured into several key components:

Header Files: Includes the necessary libraries.

Function Definitions: Functions to perform specific tasks, such as drawing the hangman.

Main Function: The core of the program where the game logic is implemented.

Header Files

#include <iostream>

#include <string>

#include <ctime>

#include <cstdlib>

iostream: Used for input and output operations.

string: Provides support for string manipulation.

ctime: Used to manipulate time and obtain current time.

cstdlib: Used for general-purpose functions, such as generating random numbers.

Function: drawHangman

void drawHangman(int lives);

The drawHangman function is responsible for visually representing the state of the hangman based on the number of lives remaining. The function takes an integer lives as input and displays the corresponding stage of the hangman drawing.

lives == 6: No hangman is drawn.

lives == 5: Head is drawn.

lives == 4: Body is added.

lives == 3: Right arm is added.

lives == 2: Left arm is added.

lives == 1: Right leg is added.

lives == 0: Left leg is added, completing the drawing.

Main Function

int main( );

The main function is where the game logic is implemented. It involves initializing the game, processing user inputs, updating the game state, and determining the game's outcome.

Steps:

Word Selection: A random word is selected from a predefined list of words.

Initialize Lives: The player starts with 6 lives.

Game Loop: The loop continues until the player either wins or loses.

Display the current state of the hangman.

Display the current state of the guessed word.

Prompt the player to guess a letter.

Update the game state based on the player's guess.

Check if the player has guessed the word or lost all lives.

Game Outcome: Display the result (win/lose) and the correct word.

Word Selection

string words[] = {"bhabin", "hangman", "program", "project", "science", "computer", "intelligence", "instruct"};

An array of words is predefined, and a random word is selected using the rand() function.

Game Loop

The loop continues while the player has lives remaining:

Input: The player guesses a letter.

Update:

If the guessed letter is in the word, the corresponding positions in the guessed word are updated.

If the guessed letter is incorrect, the number of lives decreases.

Check Win Condition: If the guessed word matches the original word, the player wins.

End of Game

If the player successfully guesses the word, a victory message is displayed.

If the player runs out of lives, the game ends with a loss, and the correct word is revealed.

Sample Output

Winning Scenario

```
 +------------------+
 |
 |
 |
 |
 |
```

Word: _____

Guess a letter: p

...

Congratulations, you won!

Losing Scenario

```
 +-------+
 |       |
 |       O
 |      /|\
 |       |
 |      / \
```

Sorry, you lost. The word was program

Conclusion

This C++ implementation of Hangman provides a simple yet effective demonstration of fundamental programming concepts such as loops, conditionals, random number generation, and user interaction. The program can be extended with additional features, such as hints, a larger word pool, or even a graphical user interface.