MOVIE GUESSING GAME

SHUBHANK PATEL
November 30,2025

Abstract

This project presents a simple Movie Guessing Game developed in C using basic programming concepts such as arrays, strings, loops, conditionals, and user input handling. The system randomly selects a movie title from a predefined list of Bollywood movies. The player must guess the movie name one letter at a time, similar to the classic Hangman game.

The program masks the movie name, tracks used letters, limits incorrect attempts, and updates the game state after each guess. The project demonstrates concepts including randomization, input validation, logical decision-making, and state-based game loops.

CONTENTS

1. Problem Definition

1.1 Overview

Guessing games are a simple yet effective way to practice logic development and user interaction in programming. This project implements a Hangman-style guessing game where the user guesses a movie title one letter at a time. Incorrect guesses reduce the number of attempts, while correct guesses reveal letters in the movie title.

1.2 Objectives

The objective of this project is to develop a movie-guessing game that:

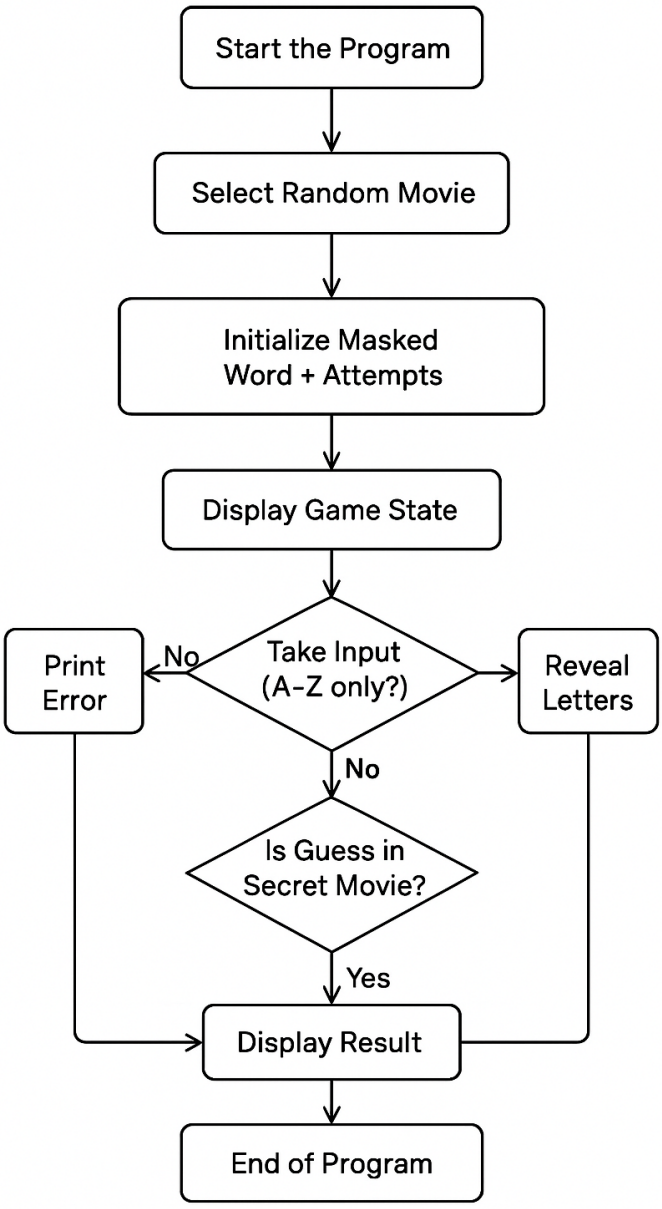
- Randomly selects a movie from a predefined list.
- Masks the letters of the selected movie using underscores.
- Accepts user guesses and validates them.
- Reveals correct letters and penalizes incorrect guesses.
- Tracks used letters.
- Determines win or loss conditions.
- Displays final results.

2. System Design

2.1 Algorithm

1. Start program.
2. Store a list of movie titles in uppercase.
3. Use randomization to select a movie from the list.
4. Initialize:
    - masked version of the movie
    - attempts counter
    - guessed letters array
5. Display current game state.
6. Take user input (one letter).
7. Validate input (A–Z only).
8. Check if the letter was already guessed.
9. Add letter to guessed list.
10. If letter is in movie: reveal it in masked word.
11. Else: decrease attempts.
12. Check win condition: masked word matches movie.
13. Check loss condition: attempts = 0.
14. Loop until game ends.
15. Display result and end program.

2.2 Flowchart



3. Implementation Details

3.1 Key Features

1. **Random Movie Selection**
   Uses `rand()` and `time()` to pick a movie from the array.

2. **Input Validation**
   Ensures user enters only A-Z characters.

3. **Masked Movie Logic**
   Shows _ for unguessed letters, preserving spaces/hyphens if present.

4. **Guess Tracking**
   Uses a boolean array to store whether each letter (A-Z) has been guessed.

5. **Attempt Counter**
   User gets a maximum of 7 incorrect attempts.

6. **Game Loop Control**
   Breaks only when player wins or loses.

7. **Clean User Interface**
   Displays attempts, used letters, and updated movie state each turn.

3.2 Code Snippets

Random movie selection

srand((unsigned int)time(NULL));

int random_index = rand() % MAX_MOVIES;

```
const char *secret_movie = movies[random_index];
```

Initialize masked movie

for (int i = 0; i < movie_len; i++)

if (secret_movie[i] == ' ' || secret_movie[i] == '-') {

```
        masked_word[i] = secret_movie[i];
    } else {
        masked_word[i] = '_';
    }
}
```

Check if guess is valid

```
guess = (char)toupper((unsigned char)guess);
if (guess < 'A' || guess > 'Z') {
   printf("Please enter a valid letter (A-Z).\n");
   continue;
}
```

Reveal correct letters

```
for (int i = 0; i < movie_len; i++) {
    if (secret_movie[i] == guess) {
        masked_word[i] = guess;
        correct = true;
    }
}
```

Handle incorrect guess

```
if (!correct) {
   attempts_left--;
   printf("Sorry, '%c' is not in the movie.\n", guess);
}
```

4. Testing & Results

4.1 Test Case 1 — Player Wins

Secret Movie: SHOLAY
Guesses: S, H, O, L, A, Y
Outcome:
✔ All letters guessed
✔ Attempts left: 5
Result: *You guessed the movie!*

4.2 Test Case 2 — Player Loses

Secret Movie: DANGAL
Guesses: X, P, Q, R, T, M, K
Outcome:
✘ Attempts exhausted
Result: *Game Over — The movie was DANGAL*

4.3 Test Case 3 — Repeated Input

Input: A
Repeated Input: A
Output: "You already guessed 'A'. Try another letter."

4.4 Test Case 4 — Invalid Input

Input: 5
Output: "Please enter a letter between A and Z."

5. Conclusion & Future Work

5.1 Conclusion

This project successfully implements a command-line Movie Guessing Game using C. It demonstrates the use of loops, conditionals, string handling, randomization, and user input validation. The game is interactive, easy to understand, and showcases logical game design fundamentals.

5.2 Future Work

Possible enhancements include:

- Add GUI-based version.
- Add hints such as genre or year.
- Allow full-movie guessing attempts.
- Store scores for multiple rounds.
- Add difficulty levels with varying attempts.

6. References

- C Standard Library Documentation (stdio.h, stdlib.h, ctype.h, string.h)
- Online resource
- Random number generation references