



# HANGMAN GAME

IN C LANGUAGE

Diya

Roll No: **CT-  
25165**



Mishal

Roll No: **CT-  
25156**



Food-Themed Words



6 Lives



C Programming

# What is Hangman?

## Objective

Guess the secret food word within limited chances.

## How to Play

1. Computer chooses a secret word
2. You guess one letter at a time
3. Correct letters reveal in position
4. Wrong letters cost 1 chance
5. Win by completing word before chances run out



## Features



Visual ASCII hangman



6 chances



Case-insensitive input



Food-themed words



# Game Rules & Flow

## Game Flow

- 1 Start → Initialize Game
- 2 While chances > 0
- 3 Show word + hangman
- 4 Get user input
- 5 Check letters
- 6 If complete → Win
- 7 Else decrement chance
- 8 If chances == 0 → Game Over

## Chances System




6 Lives - Use them wisely!


## Food Theme

 Pizza

Sandwich

 Sushi

 CUPCAKE

 Ice Cream

# Required Libraries



## stdio.h

- printf(), scanf()
- Input/Output functions
- User interaction



## stdlib.h

- rand(), srand()
- Random utilities
- Random number generation



## time.h

- time()
- Time utilities
- Seeding RNG with current time



## string.h

- strcpy(), strcmp(), strlen()
- String operations
- Word manipulation

These libraries provide essential functions for game logic, input handling, random selection, and string manipulation

# Program Start

## Code Implementation

```
int main() {  
  
    printf("\n\t\t\t\tWelcome to the Hangman Game!\n\n");  
  
    const int maxchances = 6;  
  
    printf("Guess the secret food word! You have only %d chances!\n",  
        maxchances);  
  
    // RNG seed comes next  
  
}
```

## Function Overview

The main() function serves as the entry point of our Hangman game.

## Welcome Message

Displays a friendly greeting to introduce the game and set expectations.

## Difficulty Setting

Establishes 6 as the maximum number of chances (lives) for balanced gameplay.



Next Step: Random Word Selection

# Random Word Selection

## Code Implementation

```
// Seed RNG
srand(time(NULL));

// Word bank
const char *food[] = {"pizza", "omelette", "sandwich", "icecream",
"icecream", "sushi"};

// Copy chosen word
char secretword[30];
strcpy(secretword, food[rand() % 5]);
```

## Random Number Generation

Uses current time to seed the random number generator, ensuring different words each words each run.

## Food Word Bank

Array of 5 popular food items: pizza, omelette, sandwich, ice cream, and sushi. and sushi.

## String Copying

rand() % 5 picks index 0-4, then strcpy() copies the selected word into secretword. secretword.

→ Next Step: Variable Setup

# Variable Setup

## Code Implementation

```
int chances = maxchances;
int length = strlen(secretword);
char currentword[30];
int i;
char guess;
int guessed = 0;
// Initialize current word
for (i = 0; i < length; i++) {
    currentword[i] = '_';
}
currentword[length] = '\0';
```

## Game State Variables

Track game progress: chances remaining, word length, and current guess state.

## Word Display

currentword[] holds the masked version showing underscores for unrevealed letters.

## Initialization Loop

Fills currentword with underscores to hide the secret word initially.



Next Step: Game Loop Logic

# Game Loop Start

## Code Implementation

```
while (chances > 0) {  
    printf("\nWord: %s\n", currentword);  
    Hangman(chances);  
    printf("\nEnter a letter: ");  
    scanf(" %c", &guess);  
    // processing continues...  
}
```

## Loop Condition

The game continues as long as the player has chances remaining.

## Word Display

Shows the current state of the word with revealed and hidden letters.

## User Input

Prompts for a letter guess and reads a single character from the user.



Next Step: Input Conversion

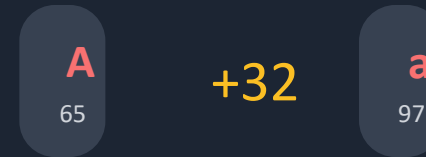


# Input Conversion (Case Handling)

## </> Code Implementation

```
if (guess >= 'A' && guess <= 'Z') {  
    guess = guess + 32; // to lowercase  
}  
  
// Alternative: tolower(guess) from <ctype.h>
```

## ↔ ASCII Conversion



## i Purpose

Makes the game case-insensitive so users can type uppercase or lowercase letters without affecting gameplay.

## 🗂 ASCII Range

A-Z: 65-90, a-z: 97-122. Adding 32 converts uppercase to lowercase.

# Checking the Guess

## Code Implementation

```
int correctguess = 0;
for (i = 0; i < length; i++) {

    if (secretword[i] == guess && currentword[i] == '_') {

        currentword[i] = guess;
        correctguess = 1;
    }
}
```

## Letter Matching

Loop through each character of the secret word to find matches with the guessed letter.

## Reveal Logic

Only reveal letters in positions that are still hidden (marked with '\_').

## Success Flag

Set correctguess flag to 1 if any letter was revealed, indicating a successful guess.



## Example: Guessing 'p' in "pizza"

Before Guess

\_ \_ \_ \_ \_

Guess 'p'

p \_ \_ \_ \_

After Reveal

p \_ \_ \_ \_

# Wrong Guess Handling

## ✖ Code Implementation

```
// Check if guess was wrong
if (correctguess == 0) {
    chances--; // reduce remaining lives

    printf("Wrong guess! Chances left: %d\n", chances);
}
```

## ♥ Visual Feedback

Each wrong guess costs one life. Players see remaining chances displayed clearly.



## ⚠ User Experience

Clear feedback helps players understand their progress and remaining remaining attempts.



Next Check: Winning Condition

# Winning Condition



## Code Implementation

```
if (strcmp(currentword, secretword) == 0) {  
  
    printf("\n 🏆 You guessed it! The word is: %s\n", secretword);  
  
    guessed = 1;  
    break;  
}
```



## String Comparison

strcmp() returns 0 when both strings are identical, meaning all letters have been revealed.



## Success Flag

Setting guessed = 1 indicates the player successfully completed the word.



## Break Statement

Exits the while loop immediately, ending the game with victory.



Exit Loop → Game Won

# GAME OVER CONDITION

## Game Over Logic

```
if (guessed == 0) {  
  
    printf("\nGAME OVER! The word was: %s\n", secretword);  
  
    Hangman(0);  
}
```

## Losing Condition

Player ran out of all 6 chances without guessing the complete word. The loop terminates when chances reach 0.

## Word Reveal

The secret food word is revealed to show the player what they missed.

## Final Hangman

Complete hangman figure is displayed (case 0) to show the final state.



Previous: Winning Condition



Next: Hangman Function Intro

# Hangman Function Intro

## Function Declaration

```
void Hangman(int chancesleft) {  
    switch (chancesleft) {  
        case 6: /* empty gallows */ break;  
        case 5: /* head */ break;  
        case 4: /* body */ break;  
        case 3: /* one arm */ break;  
        case 2: /* two arms */ break;  
        case 1: /* one leg */ break;  
        case 0: /* complete */ break;  
    }  
}
```

## Function Purpose

Display ASCII art hangman based on remaining chances (0-6). Provides visual feedback for player's progress.

## Switch Cases

7 cases representing hangman stages from empty gallows to complete figure.

## Visual Feedback

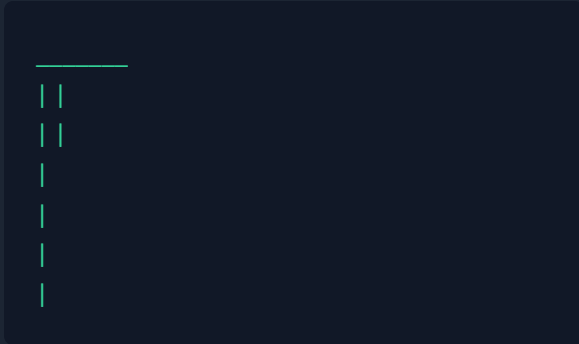
Each case builds the hangman step by step, making game progress visible to player.



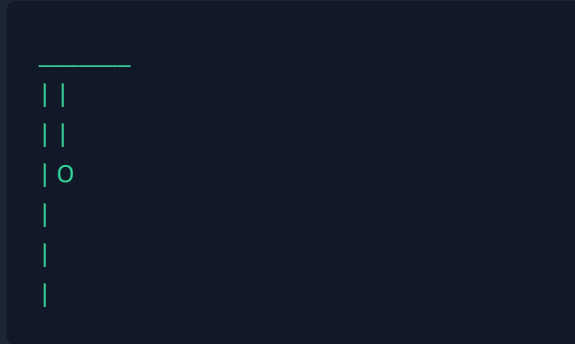
Next Step: Hangman Stages

# Hangman Stages 1-3

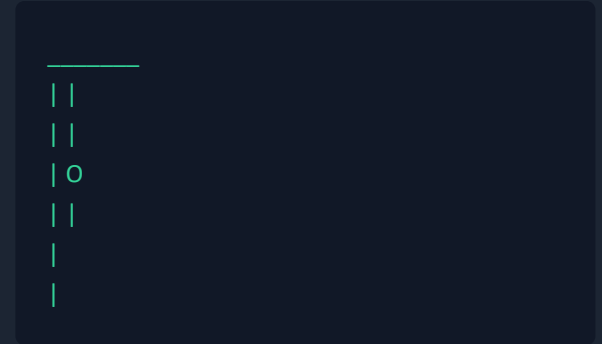
## ▶ Stage 6: Empty Gallows



## Stage 5: Head Added



## 👤 Stage 4: Body Added



Next: Arms and Legs Stages

# Hangman Stages 4-7

## Stage 3

Case 3:

```
_____  
| |  
| |  
| O  
| /|  
|  
|
```

Left Arm Added

## Stage 2

Case 2:

```
_____  
| |  
| |  
| O  
| /|\
```

Both Arms Added

## Stage 1

Case 1:

```
_____  
| |  
| |  
| O  
| /|\
```

One Leg Added

## Stage 0

Case 0:

```
_____  
| |  
| |  
| O  
| /|\
```

GAME OVER! 🎮



Final Stage: Complete Hangman



All chances exhausted



# Game Demo Example

## Example

### >\_ Terminal Simulation

```
$ Welcome to the Hangman Game!
$
Guess the secret food word! You have only 6 chances!

Word: _ _ _ _ _
$ Enter a letter:      p
Word: p _ _ _ _
$ Enter a letter:      i
Word: p _ _ z z _
$ Enter a letter:      a
$ You guessed it! The word is:      pizza
```

### 🎮 Win Scenario

Player successfully guessed "pizza" in 3 attempts, demonstrating the game flow from start to finish.

### 🏆 Key Features

Shows case-insensitive input, real-time word update, and immediate win detection when word is complete.

### ✅ Success Path

Player guesses: p → i → z → z → a. Each correct letter reveals instantly, rewarding strategic guessing.

→ Game Flow Complete

# KEY FEATURES



## Random Selection

Uses `srand(time)` and `rand()` to choose random food word from array



## Case-Insensitive Input

Automatically converts uppercase letters to lowercase using ASCII arithmetic



## Visual ASCII Feedback

Interactive hangman stages displayed for each wrong guess



## 6 Chances

Balanced difficulty with exactly 6 attempts attempts to guess the word



## Food Theme

All secret words are food items (pizza, sushi, sandwich, etc.)



## Loop Logic

Efficient while loop that continues until win or all chances exhausted

# THANK YOU!



Diya

Roll No: 165



Mishal

Roll No: 156



C Programming Project



Computer Science