**PROGRAMMING FUNDAMENTAL PROJECT:**

**PROJECT PHASE 02.**

**TOPIC:**

**THE HANGMAN GAME.**

#include <iostream>

#include <vector>

#include <string>

#include <cstdlib>

#include <ctime>

#include <algorithm> // For std::find

using namespace std;

// Function to get a random word from a predefined list

string getRandomWord() {

vector<string> words = {"apple", "banana", "programming", "hangman", "keyboard", "computer", "laptop", "project"};

int index = rand() % words.size(); // Get a random index

return words[index];

}

// Function to display the current state of the guessed word

void displayWord(const string &word, const vector<bool> &guessed) {

for (size\_t i = 0; i < word.length(); i++) {

if (guessed[i]) {

cout << word[i] << " "; // Show the guessed letter

} else {

cout << "\_ "; // Show an underscore for unguessed letters

}

}

cout << endl;

}

int main() {

srand(time(0)); // Seed the random number generator

string word = getRandomWord(); // Random word to guess

int maxAttempts = 6; // Maximum allowed incorrect guesses

int incorrectGuesses = 0; // Count of incorrect guesses

vector<bool> guessed(word.length(), false); // Track guessed letters

vector<char> wrongGuesses; // Store wrong guesses

cout << "Welcome to Hangman!" << endl;

cout << "Guess the word by entering one letter at a time." << endl;

cout << "You have " << maxAttempts << " incorrect attempts. Good luck!" << endl;

// Main game loop

while (incorrectGuesses < maxAttempts) {

cout << "\nWord to guess: ";

displayWord(word, guessed);

cout << "Wrong guesses: ";

for (char c : wrongGuesses) {

cout << c << " ";

}

cout << endl;

cout << "Enter your guess: ";

char guess;

cin >> guess;

// Convert to lowercase for case-insensitive comparison

guess = tolower(guess);

bool correct = false;

for (size\_t i = 0; i < word.length(); i++) {

if (tolower(word[i]) == guess && !guessed[i]) { // Match and not already guessed

guessed[i] = true;

correct = true;

}

}

if (correct) {

cout << "Good guess!" << endl;

} else {

// Check if the letter has already been guessed as wrong

if (find(wrongGuesses.begin(), wrongGuesses.end(), guess) == wrongGuesses.end()) {

wrongGuesses.push\_back(guess); // Add to wrong guesses

incorrectGuesses++;

cout << "Wrong guess! Attempts left: " << maxAttempts - incorrectGuesses << endl;

} else {

cout << "You already guessed that letter!" << endl;

}

}

// Check if the word is fully guessed

bool wordGuessed = true;

for (bool g : guessed) {

if (!g) {

wordGuessed = false;

break;

}

}

if (wordGuessed) {

cout << "\nCongratulations! You guessed the word: " << word << endl;

break;

}

}

// If the user runs out of attempts

if (incorrectGuesses == maxAttempts) {

cout << "\nGame over! The correct word was: " << word << endl;

}

return 0;

}

**EXPLANATION:**

Here's a **line-by-line explanation** of the Hangman game code to help you understand everything:

**Code Breakdown**

#include <iostream>

#include <vector>

#include <string>

#include <cstdlib>

#include <ctime>

#include <algorithm>

* **#include <iostream>**: Allows input/output operations using cin and cout.
* **#include <vector>**: Used to store dynamic lists, such as the word being guessed and the wrong guesses.
* **#include <string>**: Enables the use of strings for the word and guesses.
* **#include <cstdlib>**: Used for generating random numbers.
* **#include <ctime>**: Provides the current time, which seeds the random number generator for randomness.
* **#include <algorithm>**: Provides helper functions like find() for searching elements in vectors.

**Word Selection**

string getRandomWord() {

vector<string> words = {"apple", "banana", "programming", "hangman", "keyboard", "computer", "laptop", "project"};

int index = rand() % words.size(); // Get a random index

return words[index];

}

1. **vector<string> words**: Defines a list of words the program can randomly select.
2. **rand() % words.size()**: Generates a random number between 0 and words.size() - 1. This number is used as an index to select a word.
3. **return words[index]**: Returns the randomly selected word for the game.

**Displaying the Word Progress**

void displayWord(const string &word, const vector<bool> &guessed) {

for (size\_t i = 0; i < word.length(); i++) {

if (guessed[i]) {

cout << word[i] << " "; // Shows correctly guessed letters

} else {

cout << "\_ "; // Shows underscores for unguessed letters

}

}

cout << endl;

}

1. **size\_t i**: A type-safe variable for indexing through the word's characters.
2. **guessed[i]**: Checks if the i-th letter of the word has been guessed (true if guessed, false otherwise).
3. **cout << word[i]**: Prints the guessed letter.
4. **cout << "\_ "**: Displays an underscore (\_) for letters that haven’t been guessed yet.

**Main Function Setup**

srand(time(0)); // Seed the random number generator

string word = getRandomWord(); // Get a random word

int maxAttempts = 6; // Maximum allowed incorrect guesses

int incorrectGuesses = 0; // Track the number of wrong guesses

vector<bool> guessed(word.length(), false); // Track guessed letters

vector<char> wrongGuesses; // Store incorrect guesses

1. **srand(time(0))**: Seeds the random number generator to produce different random outputs each time the program runs.
2. **getRandomWord()**: Chooses the word to guess randomly from the list.
3. **maxAttempts**: Sets the maximum number of incorrect guesses allowed (6 in this case).
4. **incorrectGuesses**: Tracks the number of wrong guesses made so far.
5. **vector<bool> guessed**: A boolean array to track which letters of the word have been guessed (false for unguessed, true for guessed).
6. **vector<char> wrongGuesses**: Stores all incorrect letters guessed by the player.

**Game Loop**

while (incorrectGuesses < maxAttempts) {

cout << "\nWord to guess: ";

displayWord(word, guessed);

cout << "Wrong guesses: ";

for (char c : wrongGuesses) {

cout << c << " ";

}

cout << endl;

cout << "Enter your guess: ";

char guess;

cin >> guess;

guess = tolower(guess); // Convert to lowercase for case-insensitive matching

1. **while (incorrectGuesses < maxAttempts)**: The game continues until the user either guesses the word or uses up all their attempts.
2. **displayWord(word, guessed)**: Shows the current progress of the word.
3. **cout << "Wrong guesses"**: Displays all incorrect guesses made so far.
4. **char guess**: Takes input from the player (a single letter).
5. **tolower(guess)**: Ensures the input is in lowercase to avoid case mismatches.

**Checking the Guess**

bool correct = false;

for (size\_t i = 0; i < word.length(); i++) {

if (tolower(word[i]) == guess && !guessed[i]) {

guessed[i] = true;

correct = true;

}

}

if (correct) {

cout << "Good guess!" << endl;

} else {

if (find(wrongGuesses.begin(), wrongGuesses.end(), guess) == wrongGuesses.end()) {

wrongGuesses.push\_back(guess);

incorrectGuesses++;

cout << "Wrong guess! Attempts left: " << maxAttempts - incorrectGuesses << endl;

} else {

cout << "You already guessed that letter!" << endl;

}

}

1. **for (size\_t i = 0; i < word.length(); i++)**: Loops through each character of the word.
2. **tolower(word[i]) == guess**: Checks if the guessed letter matches the current letter of the word (case-insensitive).
3. **guessed[i] = true**: Marks the letter as guessed.
4. **correct = true**: Sets the flag to indicate the guess was correct.
5. **find(wrongGuesses.begin(), wrongGuesses.end(), guess)**: Ensures the guessed letter is not already in the list of wrong guesses.
6. **wrongGuesses.push\_back(guess)**: Adds the incorrect guess to the list.
7. **incorrectGuesses++**: Increments the count of wrong guesses.

**Checking for Victory or Loss**

bool wordGuessed = true;

for (bool g : guessed) {

if (!g) {

wordGuessed = false;

break;

}

}

if (wordGuessed) {

cout << "\nCongratulations! You guessed the word: " << word << endl;

break;

}

1. **bool wordGuessed**: A flag to track if the player has guessed the entire word.
2. **for (bool g : guessed)**: Checks each element of the guessed vector. If any are false, the word hasn’t been fully guessed yet.
3. **if (wordGuessed)**: Ends the game early if the word is completely guessed.

**End of Game**

if (incorrectGuesses == maxAttempts) {

cout << "\nGame over! The correct word was: " << word << endl;

}

* If the player runs out of attempts, the game ends, and the correct word is revealed.

**Key Features Recap**

* **Random Word Selection**: Ensures variety in each playthrough.
* **Tracking Progress**: Displays guessed letters and underscores for the remaining ones.
* **Wrong Guess Handling**: Prevents repeated wrong guesses and keeps track of incorrect attempts.
* **Win/Lose Conditions**: Ends the game when the player guesses the word or runs out of attempts.

Let me know if you have any specific doubts or want further clarification! 😊