Solution Review Hangman Game

In this lesson, you will see the step by step solution review of the hangman game given to you in the previous lesson.

We'll cover the following

- Solution
- Solution review
  - Step 1: Initialize state variables
  - Step 2: Display game details
  - Step 3: Select secret word randomly
  - Step 4: Display blanks for every letter in a secret word
  - Step 5: Take a letter from the user and check whether the player guessed it right
  - Step 6: Display man on a gallow
  - Step 7: Player win/loss

Solution

Press the **RUN** button and see the output!

```
#include<iostream>

#include<cstdlib>

#include<ctime>

#include <string>

using namespace std;

void displayGameDetails(int maxTries);
string chooseSecretWord();
void displayWord(string word, int length);
int isGuessTrue(string secretWord, char guessWord[], char letter);
void displayMan(int remainingGuess);


int main() {
    int maxTries = 5;
    int remainingTries = 5;
```

```
    int remainingTries = 5;
    char guessLetter;
    string secretWord;
    int secretWordLength;

    displayGameDetails(maxTries);

    secretWord = chooseSecretWord();
    secretWordLength = secretWord.length();

    char guessWord[secretWordLength];
    for (int i = 0; i < secretWordLength; i++) {
        guessWord[i] = '-';
    }
    cout << "Your guess word is:";
    displayWord(guessWord, secretWordLength);

    while (remainingTries != 0) {

        cout << "Enter your guess letter:" << endl;
        cin >> guessLetter;

        int guess = isGuessTrue(secretWord, guessWord, guessLetter);

        if (guess == 0) {
            remainingTries--;
            cout << "\nWhoops! that letter is not present in the word" << endl;
            displayMan(remainingTries);
        }
        if (guess == 1) {
            cout << "\nYay! You have found the letter" << endl;
        }
        if (guess == 2) {
            cout << "\nYou have already guessed this letter. Try something else!" << endl;
        }
        cout << "You can have " << remainingTries << " more wrong attempts" << endl;
        cout << "Your guess word is:";
        displayWord(guessWord, secretWordLength);
        cout << endl;

        if (secretWord == guessWord) {
            cout << "\nCongratulation! You won." << endl;
            return 0;
        }

    }
    if (secretWord != guessWord){
    cout << "\nToo many Guesses! You have been hanged." << endl;
    cout << "\nThe secret word was: ";
    displayWord(secretWord, secretWordLength);
    }
}

void displayGameDetails(int maxTries){
    cout << "\n"
    "        .-------------------------------------------------------------------------------.\n"
    "        |                                                                               |\n"
    "        |     _      _                                                                  |\n"
    "        |     | |    | |   __ _    _ __      __ _        /\\/\\      __ _    _ __        |\n"
    "        |     | |----| |  / _  |  |  _ \\    / _` |      /    \\    / _  |  |  _  \\       |\n"
    "        |     | |----| | | (_| |  | | | |  | (_| |     / /\\/\\ |  | (_| |  | | | |       |\n"
    "        |     |_|    |_|  \\_ _ |  |_| |_|   \\__, |     \\/    \\/   \\_ _ |  |_| |_|      \n"
    "        |                                     |___/                                     |\n"
    "        .-------------------------------------------------------------------------------.\n"
```

```
    cout << "The purpose of this game is to guess an animal name, secretly chosen by the applicatio
    cout << "You have to guess one letter at a time and you can have " << maxTries << " wrong atte
    cout << "Enter a lower-case letter and don't forget to enter key after each guess\n\n";
    cout << "Let's play the game!\n\n";
}

string chooseSecretWord() {
    srand(time(NULL));

    string animals[] = {"puppy","turtle","rabbit","raccoon","kitten","hamster",
        "sheep","turkey","chicken","horse","chimpanzee","kangaroo","koala",
        "elephant","leopard","hippopotamus","giraffe","crocodile","alligator",
        "hedgehog"
    };

    int randomIndex = (rand() % 20);
    string word = animals[randomIndex];
    return word;
}

void displayWord(string word, int length) {
    for (int i = 0; i < length; i++) {

        cout << word[i];
    }
    cout << endl;
}

int isGuessTrue(string secretWord, char guessWord[], char letter) {
    int flag = 0;
    for (int i = 0; i < secretWord.length(); i++) {
        if (secretWord[i] == letter) {
            if (guessWord[i] == secretWord[i]) {
                flag = 2;
            } else {
                guessWord[i] = secretWord[i];
                flag = 1;
            }
        }
    }
    return flag;
}

void displayMan(int remainingGuess) {

    string part[4];
    switch (remainingGuess) {
    case 0:
        part[3] = "|";
    case 1:
        part[2] = "/|\\";
    case 2:
        part[1] = "/|\\";
    case 3:
        part[0] = "( )";
        break;
    }

    cout << "                                                ----------------\n";
    cout << "                                                    |       " << part[3] << endl;
    cout << "                                                    |       " << part[3] << endl;
```

```
    cout << "                                                |      " << part[0] << endl;
    cout << "                                                |      " << part[1] << endl;
    cout << "                                                |      " << part[2] << endl;
    cout << "                                                |           \n"
    "                                              ---------------   \n";
}
```

Solution review

Step 1: Initialize state variables

First, we must initialize some variables to store the game statistics.

```
    int maxTries = 5;
    int remainingTries = 5;
    char guessLetter;
    string secretWord;
    int secretWordLength;
```

- `maxTries` indicates how much time the player can guess the wrong letter.
- `remainingTries` keeps track of the number of wrong tries that the player has left.
- `guessLetter` stores the character guessed by the user.

Step 2: Display game details

Then, we call the `displayGameDetails(maxTries)` function to display some necessary details about the game. It just uses the `cout` statement to print some instructions on the terminal.

Step 3: Select secret word randomly

We call the `chooseSecretWord()` to make an array of words and select the secret word randomly from the given array. In this function:

- First, we create an array of strings called `animals[]` and store all the secret words in it.
- To choose the word randomly from an array, we use the `rand()` and `srand()` functions defined in the `<cstlib>` library. We also use the `time` function defined in the `<ctime>` library. By using these built-in functions, we can select a random index from an array and then return the word at that index.

- `rand()` takes nothing in input and generates a random number from **0 to 2147483647**. `rand()` does not generate truly random numbers. Instead, it generates a pseudo-random number, which means each time you compile the program, you get the same sequence of numbers. We can use the modulus operator to scale the random numbers within the given range.

- To generate a different sequence of random numbers each time the program executes, we use the `srand()` function. `srand()` seeds the random number generator so that each time we start in a different place. `srand()` takes seed in its input. We only need to call `srand() once in a program`. The most common way to `seed` the random number is to use the current time in `srand()` input. We use the `time()` function defined in the `ctime` library to know the current time. The `time()` function takes `NULL` in it's input

```
#include<cstdlib>

#include<ctime>

string chooseSecretWord() {

    string animals[] =
    {"puppy","turtle","rabbit",
     "raccoon","kitten","hamster",
     "sheep","turkey","chicken",
     "horse","chimpanzee","kangaroo",
     "koala", "elephant","leopard","hippopotamus",
     "giraffe","crocodile","alligator",
     "hedgehog"
    };

    srand(time(NULL));
    int randomIndex = (rand() % 20);
    string word = animals[randomIndex];
    return word;
}
```

Step 4: Display blanks for every letter in a secret word

Now that we have selected the secret word, we want to display the number of letters present in the word by using the blanks `_`.

First, we find the length of that secret word by using `.length()` property of the string. We initialize an array of characters `guessWord` whose length is equal to the

length of the secret word. Initially, we fill all the array letters with the blanks `-` and use the `displayWord()` function to print its letters on the console.

```
secretWordLength = secretWord.length();

    char guessWord[secretWordLength];
    for (int i = 0; i < secretWordLength; i++) {
        guessWord[i] = '-';
    }
    cout << "Your guess word is:";
    displayWord(guessWord, secretWordLength);
```

Step 5: Take a letter from the user and check whether the player guessed it right

We need to check the following conditions:

- If the player guesses a correct letter, that is present in the secret word. Then we have to reveal that letter in the correct position.
- If the user guesses a letter that has already been revealed before, then do nothing.
- If the user guesses the wrong letter, subtract the number of attempts that the player can have.

To implement this logic, we keep executing the loop statements until the user guesses the correct word or until a certain number of incorrect guesses. We take input from the user and then call the `isGuessTrue()` function to check the above-mentioned conditions.

```
int isGuessTrue(string secretWord, char guessWord[], char letter) {
    int flag = 0;
    for (int i = 0; i < secretWord.length(); i++) {
        if (secretWord[i] == letter) {
            if (guessWord[i] == secretWord[i]) {
                flag = 2;
            } else {
                guessWord[i] = secretWord[i];
                flag = 1;
            }
```

```
        }
    }

    return flag;
}
```

- If the `isGuess()` function **returns 0**, it means the player has guessed the wrong letter and our remaining tries are subtracted by one. We also display some parts of the man on a gallows by calling the `displayMan()` function.
- If the player has guessed the correct letter the `isGuess()` function reveals the word at that index and **returns 1** to the calling point.
- If the `isGuess()` function **returns 2**, it means the letter has been revealed before.

```
        int guess = isGuessTrue(secretWord, guessWord, guessLetter);

        if (guess == 0) {
            remainingTries--;
            cout << "\nWhoops! that letter is not present in the word" << endl
;
            displayMan(remainingTries);
        }
        if (guess == 1) {
            cout << "\nYay! You have found the letter" << endl;
        }
        if (guess == 2) {
            cout << "\nYou have already guessed this letter. Try something els
e!" << endl;
       }

    }
```

Step 6: Display man on a gallow

We need to draw a man on the gallow in five different steps. For this, we call the `displayMan` function, which takes the number of remaining wrong tries that a player can have in its input and uses the `switch()` statement along with `cout` to display the hangman accordingly.

```
void displayMan(int remainingGuess) {

    string part[4];
```

```
    switch (remainingGuess) {
    case 0:

        part[3] = "|";
    case 1:
        part[2] = "/|\\";
    case 2:
        part[1] = "/|\\";
    case 3:
        part[0] = "( )";
        break;
    }



cout << "                                        ----------------\n";
    cout << "                                                |       " << par
t[3] << endl;
    cout << "                                                |       " << par
t[3] << endl;
    cout << "                                                |      " << part
[0] << endl;
    cout << "                                                |      " << part
[1] << endl;
    cout << "                                                |      " << part
[2] << endl;
    cout << "                                                |           \n"
    "                                         ----------------   \n";
}
```

Step 7: Player win/loss

While iterating through the loop statements, we keep checking whether the guess word is equal to the secret word at the end. If yes, we return 0 to the calling point to terminate the application.

```
if (secretWord == guessWord) {
      cout << "\nCongratulation! You won." << endl;
      return 0;
}
```

If we come out of the loop and the secret word is still not equal to the guess word then, the player loses the game!!

```
    if (secretWord != guessWord){
        cout << "\nToo many Guesses! You have been hanged." << endl;

        cout << "\nThe secret word was: ";
        displayWord(secretWord, secretWordLength);
    }
```

You can play around with the code and make any changes to it to understand things deeper.

The next chapter contains the concluding remarks for this course.