Project 1 - Hangman

Miguel Juarez

February 2024

1 Introduction

This program was written to play a concise version of the game Hangman. Without the figure being drawn, players can still experience the game with lives in place of the well-known hanging man. This project took a lot more planning than coding but during the development process, I learned what I needed to accomplish but was not able to implement at the time of the final version. I have, to the best of my ability, completed a sort of working game that allows for player interaction and how the final product will result.

2 How To Use the Program

This program has very simple steps that the user can also follow along with easy readability and user-friendliness. Begin by running the code, the user will be displayed with a welcome message and a list of rules to follow about how to play the game. The user is then asked to pick a number from 1 - 10 that will determine the word that they will be trying to guess. A prompt then appears for the user to input their name that will hold their name in a file to later add points upon completion of the word.

3 Program Design

The design for this program requires several C++ libraries to be included for the game to operate such as a format library, RNG Library, input and output library, and reading and writing to a file. I was not able to find a reason to use ¡cmath¿ so I left it out completely. The program in its entirety runs inside the **int main(int argc, char** argv)** function and returns a 0; when the execution is successful. The program runs and begins to declare the initial variables that are needed to run the rest of the lines of code including holding the user name, the word they chose, and also the amount of lives they have left before a game is over.

The game displays a friendly way of introducing any new and returning players to hangman and after the user is finished reading, they are prompted to

choose a random number that will dictate the word that they will be trying to solve with increasing difficulty. The number represents a level with 10 being the hardest word to try and guess compared to level 1 with the easiest word. The player is given freedom to choose which word to guess allowing them to have a fun time and not become overwhelmed if they are not able to solve a difficult word.

```
//Choose a random number from 1 to 10
int chWrd = (rand()%10) + 1;
cout<<"Choose a level from 1 - 10:";
cin>>chWrd;
```

These lines of code function through the **¡cstdlib¿**and **¡ctime¿** which operate the random seed generator using the time library to generate numbers from a desired range. The range is set using **(rand() mod 10) + 1;** which mods a pseudo random number by 10 ensuring that the number falls between 0 and 9 and adding 1 makes the random number to be between 1 and 10.

The following lines of code determine the level and the word that the player will be attempting to solve. This switch case follows 10 cases that, based on the number chosen by the player, will be able to move onto the next "screen".

```
//Set word to string
switch(chWrd){
    case 1: gsWrd = "CAR"; break;
        case 2: gsWrd = "TREE"; break;
        case 3: gsWrd = "MONKEY"; break;
        case 4: gsWrd = "LINUX"; break;
        case 5: gsWrd = "COMPUTER"; break;
        case 6: gsWrd = "WEREWOLF"; break;
        case 7: gsWrd = "INFLATION"; break;
        case 8: gsWrd = "AUTOMOBILE"; break;
        case 9: gsWrd = "RELATIONSHIP"; break;
        case 10: gsWrd = "QUINTESSENTIAL"; break;
}
```

This switch case works by setting a variable defined as a **string** data type that will set the value of the variable equal to the word given in every separate case depending on the number chosen by the player. The case breaks out once the variable is assigned and moves on and is stored to be solved later in the game.

4 Read and Write to a File

Some variables that were declared in the beginning aren't used until mid-way through the program. The string data types will hold the player name that will be written to a file, this will also hold the player's name for archive purposes. There is also a point system that is a work in progress that will, like the user name, be written into a file, and beside it will hold the player's name and the

number of points they have and keep track of how many times they have solved the word with 10 points being a **const int** that will be displayed at the end of the game.

```
//Name to keep score
cout<<"Enter your name: ";
cin>>usrNm;
//Check if user inputs a name
//Create text file to save words
ofstream wrdBnk("nameNpoints.txt");
while (usrNm.empty()){
    cout<<"Invalid, please input your name again: ";
    cin>>usrNm;
}
//Write to the file
wrdBnk << usrNm;
wrdBnk.close();
```

This section of the program will ask the user for their name and store their name in the **usrNm** string variable that will be equal to the name they wrote in. We then create a text file to save the string stored in the variable as we will be using it to hold the player's points if they guess the word correctly. If the user does not input anything when asked for their name, they will receive an error message that asks them to please input their name again. This will loop until the user eventually writes their name and will then be written to the file, and closed so that there are no overwrite issues.

5 Game play

We then move to the next piece of code that will be executed which is where the player will be playing the actual game until 1 of two conditions are met that will end the game, promptly to what the user has done. These conditions will end the game if the player has correctly guessed the word with more than 0 lives left and will be awarded 10 points that will be stored in the file they originally wrote to store their name.

```
//If the player wins, their points are added to the file
ifstream rdWrds("nameNpoints.txt");
while (rdWrds >> usrNm) {
    cout << usrNm << " has won " << WIN_POINTS << endl;
}
//Update the file
wrdBnk << usrNm << " " << WIN_POINTS << " points."<<endl;
```

The file **nameNpoints.txt** is opened up after being closed initially when the player wrote their name in the file, to not only read and then update as well.

The file is updated with the display showing the user name, and the amount of points they have earned. The WIN POINTS constant variable will only be displayed and never updated because of its variable declaration.

The game begins with the following code:

```
//Print out word and dashes for word characters
cout<<"Current word: ";
for (int i=0; i<gsWrd.length(); i++){
    cout<<"- ";
}
cout<<endl;
```

This will read the length of the word int he gsWrd variable, which is the word chosen by difficulty by the player. The word is run through a for loop that will count each letter inside of the string and for each of those letters, a dash will be displayed to show the number of letters needed to win the game.

The most important part of the game lies within an if statement that checks for any condition of the switch statement and beings to execute within the block of code.

```
//Start of Word Solving
if (chWrd == 1 || chWrd == 2 || chWrd == 3 || ...etc ) {
        //Check player lives, deduct lives, check user letters
    }
```

If this statement is true it will execute the following, begin with a while loop that executes as long as the player has over 0 lives. If the player has no more than lives, the while loop condition will become false and a game-over message will display. The while loop loops and checks for letters inside of the string and makes sure that they match the string. If they match, they will not be deducted a life but rather will continue until they have no more.

```
//Check if the player still has lives to play
    while (lives > 0) {
        cout << "Guess a letter: ";
        cin >> usrLtr;
        //Convert user letter to capital
        usrLtr = toupper(usrLtr);

        // Check if the guessed letter is in the word
        bool crLtr = false;
        for (int i=1; i<gsWrd.length(); i++){
            //Access a character inside the given string
            //This case will be Word to Guess
            if (usrLtr == gsWrd[i]) {
                crLtr = true;
                break;
            }
```

```
}
//Update guessed word depending on bool expression
if (crLtr) {
    gsWrd += usrLtr;   //Correct Letter chosen
} else {
    //Decrement lives if user chooses wrong letter
    --lives;
    cout <<"Incorrect. Lives remaining: "<<lives<< endl;
}
```

This is what is inside the if statement and is what dictates continuity within the game. While some of the parts are not complete, this is as best as I was able to accomplish. I was able to learn about finding the length of a string and accessing components inside a string. I was not able to get my boolean statements to update the dashes within the **Guess word:** section but will be working on how to make it happen.

6 Results

The resulting gameplay should look something like this when the player wants to begin playing:

```
Welcome to HangMan!
Rules:
1. Do not repeat the same letter
2. Solve the mystery word
3. Have fun!
--------------------------------
Choose a level from 1 - 10: 3
```

The player chooses level **3**. The word to guess is **LINUX**.

```
Enter your name: John
```

The player enters **John** as their name and it is stored in the **nameNpoints** text file.

```
Current word: - - - - - -
Guess a letter: i
```

The player is prompted to choose any letter in the alphabet and proceeds to enter the letter **i**. The letter is correct and does not take away a life.

```
Guess a letter: s
Incorrect. Lives remaining: 4
```

The player enters the letter **s** which is not in the word LINUX. The program detects that the letter is not in the **string gsWrd = "LINUX"** and is deducted a life for the incorrect letter. The game continues until the player solves the word, or in this case, does not, and is prompted to the game over screen.

```
Game over! You ran out of lives. The word was: LINUX
John has won no points.
```

The player's name is read back in from the file and promoted with no points won in this case. The word is revealed and the game ends after this screen.

7 References

1. https://cplusplus.com/
2. Gaddis, Tony, et al. Starting out with C++. Pearson Education, Inc., 2020.
3. R, Gourav. "C++ Program to Find the Length of a String." Scaler Topics, Scaler Topics, 29 Sept. 2022, www.scaler.com/topics/string-length-cpp/.