William George
October 18, 2014
wordGuessDesign Week 3
CS 161

1. (10) Start a design based on what we know about user input, variables, if statements, and loops for a word guessing game. What would your word guessing game behave like? You can think of this like a hangman game, a Wheel of Fortune game, or other similar word guessing game. Your response to this should contain some sort of **explanation or drawing** for what you would create for a word guessing game if it were completely up to you.
(**Note**: if you have not learned anything about strings yet, then you can simply write directions such as “get word as input”, “check that word is valid”, “get other user's letter guess as input”, “check if letter is in word”, ...)

My goal is to create a game that will ask the first player to input a word. It will tell the player the number of letters in the world and the number of guesses the player will get. The number of guesses will be 2 times the length of the total word.

While the player still has guesses left or has not guessed correct already the program will ask the second player to guess letter by letter to see if each letter is in the word. After each guess the program will either say yes you got it right then output the character, the position of the letter and a representation of the word guessed so far; or it will say no. The guess counter will increment by one. The program will then ask player two to solve and player two can then enter a complete word to see if it matches the secret word. Each solve attempt will cost the player a guess as well.

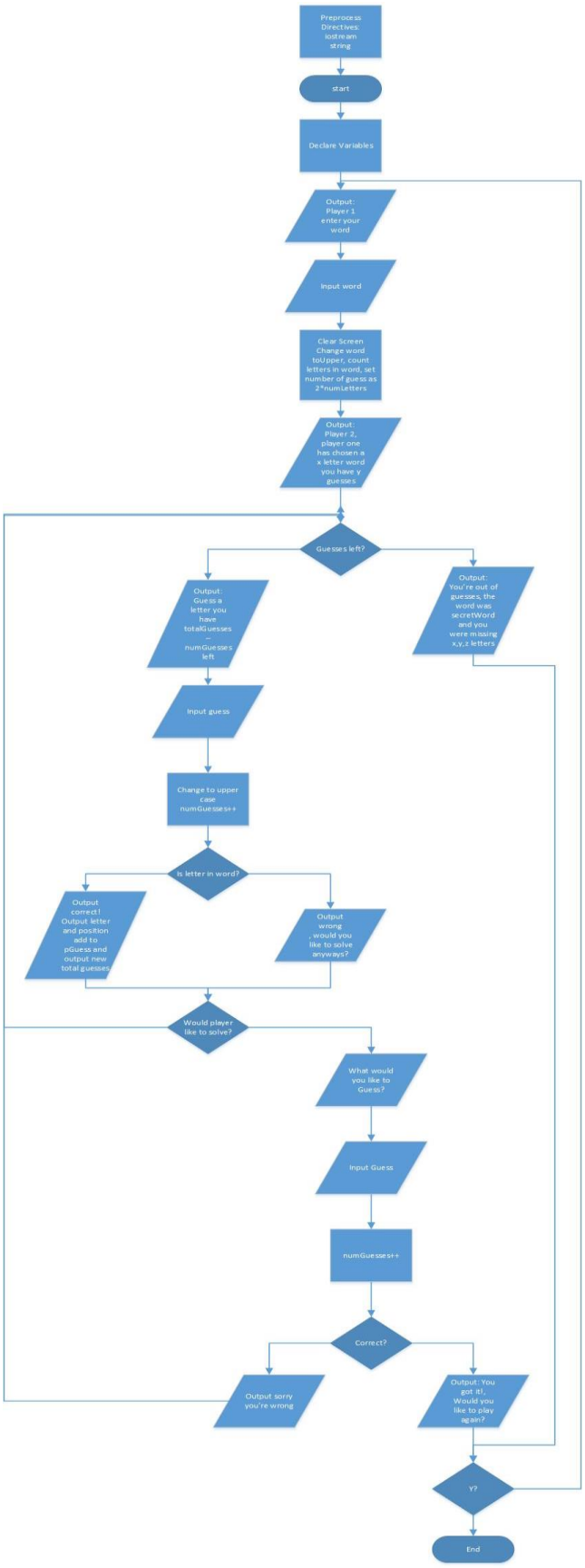
If it doesn’t match the screen will clear and the player will be prompted to enter another letter.

Once the player is out of guesses the program will tell player two the word, the letters he/she got correct and the letters he/she did not get then ask if they would like to play again.

Below is some Pseudocode and a flow chart describing how the program would work:

```
Preprocess directives: iostream , string, iomanip
Start

const int SIZE = 20;
char secretWord[SIZE];
char playerGuess;
char pGuess[SIZE];
string sWord = secretWord;
int numGuesses = 0;
int numLetters = 0;
int totalGuess = 0;
char p2Solve;
char solve;
char again;
int lettersCorrect;

do
        Output<<”Player one please give me one SECRET word that is less than 20 letters long: ” endl;
        Input>>secretWord;

        Validate secretWord is all letters and change word to single case – toUpper, also make sure word is less than 20 letters long
```

```
Count letters in word
If(isalpha(secretWord.at(i)))
        numLetters = numLetters + 1;
        totalGuess = 2 * numLetters;
system("cls");
Output: "Player two: Player One has chosen a numLetters word you have totalGuesses guesses to choose the word.

While (p2Solve != secretWord && numGuesses <= totalGuess)

                System("cls");

                Output: Player 2 Please guess a letter:
                Input: playerGuess;
                playerGuess toUpper

                For loop – for(int x =0; x <sWord.length(); x++)
                If (playerGuess == sWord.at(x))
                        Output: You guessed correct there is a playerGuess!
                        Output letter and position(s)
                        Add letter and position to variable = pGuess
                        lettersCorrect++
                        Output new pGuess
                Else
                        Sorry you're wrong, try again. Unless you'd like to solve?
                numGuesses ++
                Output: Press Y if you'd like to solve, if you do choose to solve it will cost you guess.
                Input: solve
                While (solve == Y || solve == y)
                        Output: What is your guess?
                        Input psolve;
                        Compare p2Solve to secret word;
                        numGuesses++
                        If p2Solve = secretWord
                                Output: You got it!!
                        Else: Sorry, try again!
                Output: Press Enter to enter another value
                cin.get();
                cin.get();


        if (p2Solve == secretWord)
                Output: You got it! Great Job! The word was secretWord. Press Y if you'd like to play again
        Else
                lettersMissing = find_first_not_of pGuess == secretWord
                Output: Sorry you're all out of guesses. The secret word was secretWord you got lettersCorrect letters correct
                but missed lettersMissing. Press Y if you would you like to play again?
        Input again;
While (again ==y || again == Y)
Return 0;
END
```


Preprocess Directives: iostream string
start
Declare Variables
Output: Player 1 enter your word
Input word
Clear Screen Change word toUpper, count letters in word, set number of guess as 2*numLetters
Output: Player 2, player one has chosen a x letter word you have y guesses
Guesses left?
Output: You're out of guesses, the word was secretWord and you were missing x,y,z letters
Output: Guess a letter you have totalGuesses - numGuesses left
Input guess
Change to upper case numGuesses++
Is letter in word?
Output correct! Output letter and position add to pGuess and output new total guesses
Output wrong , would you like to solve anyways?
Would player like to solve?
What would you like to Guess?
Input Guess
numGuesses++
Correct?
Output sorry you're wrong
Output: You got it!, Would you like to play again?
Y?
End

2. (30) You will receive more specific requirements later, this design is based on your own ideas to start with, so if these following requirements were imposed upon your program and design, then how might you implement each of the following:

a. (05) it needs to store player 1's "secret word",

To store the player's secret word you would has a variable named something like secretWord which would store the players input. This can be taken as a string variable but must be converted to a char input. To take this input you would likely just need a cin statement.

b. (05) the "secret word" will only be allowed to contain letters (at least for now),

You need to include some sort of validation here, some possibilities were discussed in the previous projects. You need to validate the word has letters and not numbers or characters as well as the length of the word. Also to simplify things later on the word should be converted to all upper or lower case characters.

The program will also output the total number of letters to player 2 and calculate the total number of guesses as 2 two times the number of letters in the secret word.

c. (05) the program will repeatedly ask for single letters from player 2,

I will use a 'while' loop to repeatedly ask for letter.
The letter will be turned into an upper case letter here to help keep the comparison valid.
Then a 'for' loop to compare the letters to the secret word.

Then using an 'if' statement, the program will output:
Whether the guess was correct,
The position of the letter in the word
And a word with either spaces or dashes representing the current status of all the guesses in comparison to the total word.

d. (05) the program will need to show correct letter guesses and the word so far based on those correct guesses,

If the letter is correct then the program will output the number of letters correct and the letters that match. Also if letter repeats in a word the program will search for all repeats of the letter and output each position of the letter as well as a current state of the entire word. After each guess the program will ask if the player would like to solve and a solve counts as another guess used.

e. (05) player two will only get so many guesses (how will you decide this?),

The total number of guesses will be 2 times the total length of the word. Player 2 solves and letter guesses will count as a guess.

f. (05) when done, the program will need to show the secret word, list which letters were missing still, and ask if the players want to play again.

The program will show the secret word and whether the player got it correct, then use the find_first_not_of command to output the letters missed by player 2. Final the entire program will be in a do-while loop to ask if the player would like to play again.