**Project 1**

**Hangman**

**CSC-5**

**Section 40375**

**02/05/14**

**Jennifer Clark**

**Introduction**

Hangman is a fairly well-known game but it is also easy to learn. This is a word game, with the objective being to guess a word by choosing letters. The catch is that the player must guess the word within six wrong answers or you reach Hangman, which means that the man whose life has been on the line for this game is now dead in a hangman’s noose.

Initially the player is shown blank spaces equivalent to the number of letters in the word. As letters are guessed, any correct letters will replace the blanks in the word. As long as the player can guess the word within six wrong letter-guesses, they win!

In this version of the game if the player accidently guess an incorrect letter multiple times it still counts against their score.

**Development Summary**

Lines of Code: 173 (292 including code for primitive graphics)

Number of Variables: 13

IDE: Netbeans

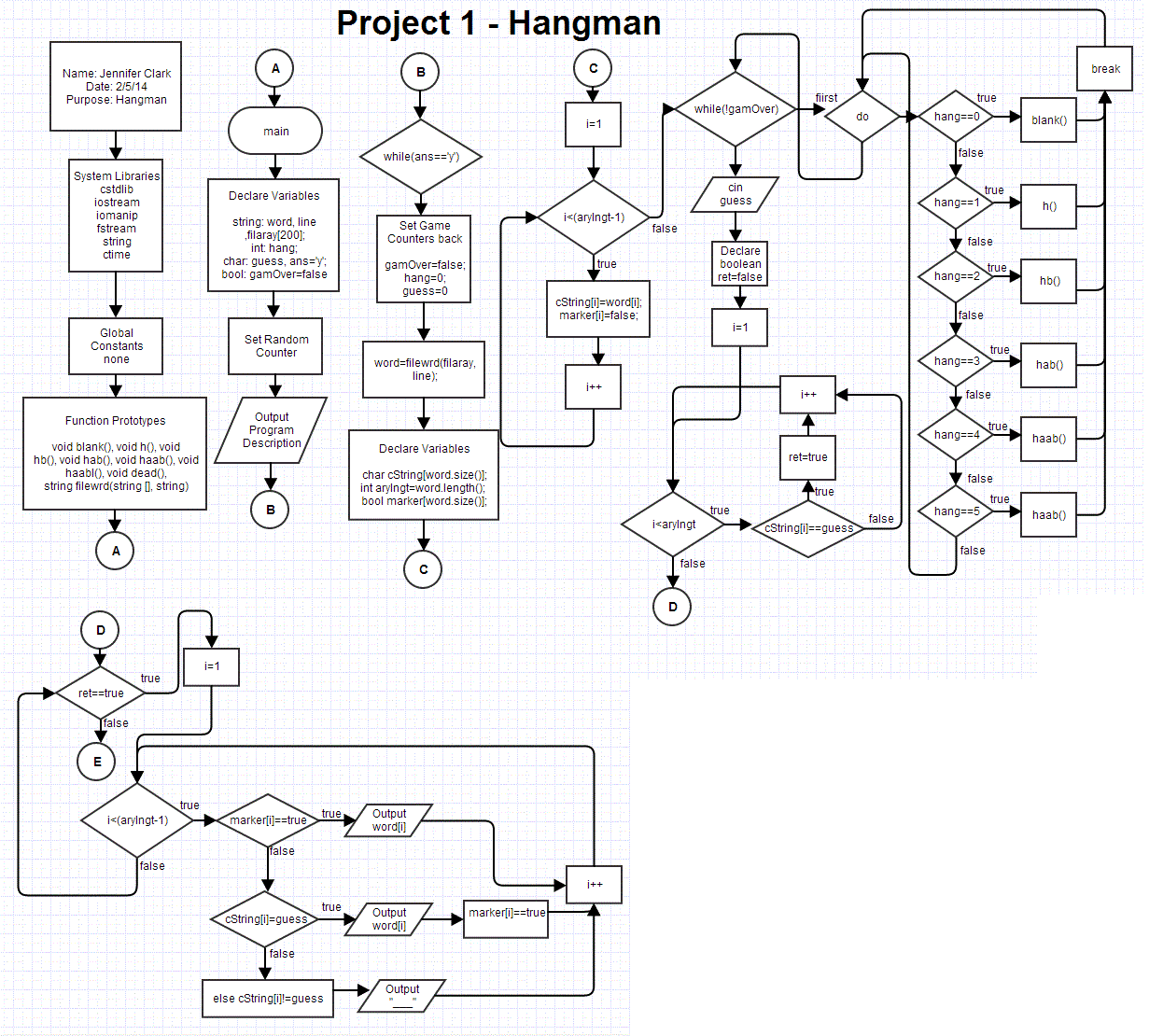
This game is simple enough to learn, but it involved a lot of complex logic in order to function properly. The initial program design sketch was lacking in detail when it came time to code, and in order to proceed several Boolean variables and loops were needed.

The Internet was a decent resource in helping me get ideas but I couldn’t find much that was specific to the task that I had at hand and also low-level enough for me to understand as a first-time programmer.

The most useful resource was review of past programs taught and coded from earlier on in the class. We’ve covered a lot of topics, and these programs provided a wealth of information when the general method was understood but the exact coding syntax was forgotten.

The program has practically unlimited opportunities for improvement. One of the biggest would be to provide a running total of the incorrectly guessed letters, because it is easy to forget a wrong guess, and due to the large graphic display for each guess it is irritating to have to scroll up to recall a previous guess. It would be an improvement in program readability to put more of the code into separate functions. There could also be different levels of difficulty which would pull from different word files, and an improvement that I would like to make is to have an option to read in Spanish words, so my fiancé can have more fun playing the game.

**Flowchart**



**Pseudo Code**

*Declare variables & set the random counter*

*Begin game loop:*

*Reset counters*

*Set ‘word’ equal to the filewrd function*

*Create a string to read in the word as an array of characters*

*Create an integer to determine the array length*

*Create a marker to eventually determine if a character in a word should be hidden*

*For an integer starting at 0, while the integer is less than the array length (minus 1):*

*The array of characters is equal to the number of characters in the word*

*The marker for each character in the array is set to be hidden (‘false’)*

*Begin the guessing loop (while game is not over):*

*If the number of wrong guesses is 0, output a certain graphic once*

*If the number of wrong guesses is 1, output a certain graphic once*

*If the number of wrong guesses is 2, output a certain graphic once*

*If the number of wrong guesses is 3, output a certain graphic once*

*If the number of wrong guesses is 4, output a certain graphic once*

*If the number of wrong guesses is 5, output a certain graphic once*

*Output the hidden word BELOW the gallows graphic*

*Request and receive input in the form of “guess”*

*Initialize a Boolean “ret” to “false”*

*For an integer starting at 0, while the integer is less than the array length:*

*If the character in the word string matches the “guess”, change ret to “true”*

*If “ret” is “true”, then for an integer starting at 0, while the integer is less than the array length (minus 1):*

*If the marker is “true” for a character, output that character*

*If the cString marker is “true” for a character, also output that character*

*Else output an underscore mark to signify an un-guessed character*

*If “ret” is not “true”, increment the “hang” counter by one and tell the player the letter is not found in the word*

*Output the number from the “hang” counter to tell the player how many guesses they have used.*

*Set the gamOver Boolean to “true”*

*If any character in the array has been marked as “false”, change the gamOver Boolean back to “false”*

*If the “hang” counter is greater than 5, set the gamOver Boolean to “true”*

*If gamOver is “true” and the “hang” counter is 6:*

*Show the “dead()” function graphic and tell the user the game is over*

*Output the word to the user*

*If gamOver is “true”*

*Ask the user if they would like to play again and receive the input for the answer*

*If the answer is not ‘y’, end the program*

*If the answer is still ‘y’, loop the program*

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Name** | **Location** | **Description** |
| string | word | main | The random word returned from function filewrd(filaray,line) |
|  | line | main, filewrd(string [],string) | One word pulled from the random\_word\_list.dat file |
|  | filaray[loop] | main, filewrd(string [],string) | One word pulled from the random\_word\_list.dat file, entered into an array |
| int | randnum | filewrd(string [],string) | A random number in the domain of the loop count. |
|  | hang | main | Counter for the number of incorrect letter guesses |
|  | arylngt | main | the number of characters in the word |
| short | loop | filewrd(string [],string) | Counter for input of strings from random\_word\_list.dat |
| char | guess | main | Input of a letter guess- compared against the letter character of each letter in cString[i] for matches |
|  | ans | main | Input to play again or exit the program |
|  | cString[word.size()] | main | The character array with the domain being the size of the word string |
| bool | gamOver | main | Boolean to determine if the game is over or not |
|  | marker[word.size()] | main | The marker to determine whether a character in the word should be displayed on screen or hidden |
|  | ret | main | Changes to true if a character in the word is guessed correctly |

**References**

Pearson Custom Computer Science for RCC (Savitch 8th Edition)

<https://github.com/Riverside-City-College-Computer-Science/CSC5_Winter_2014_40375>

<https://github.com/1Asenath/jc1929709>