**Project 2**

**Hangman**

**Revisited**

**CSC-5**

**Section 40375**

**02/13/14**

**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: 374

Number of Variables: 13

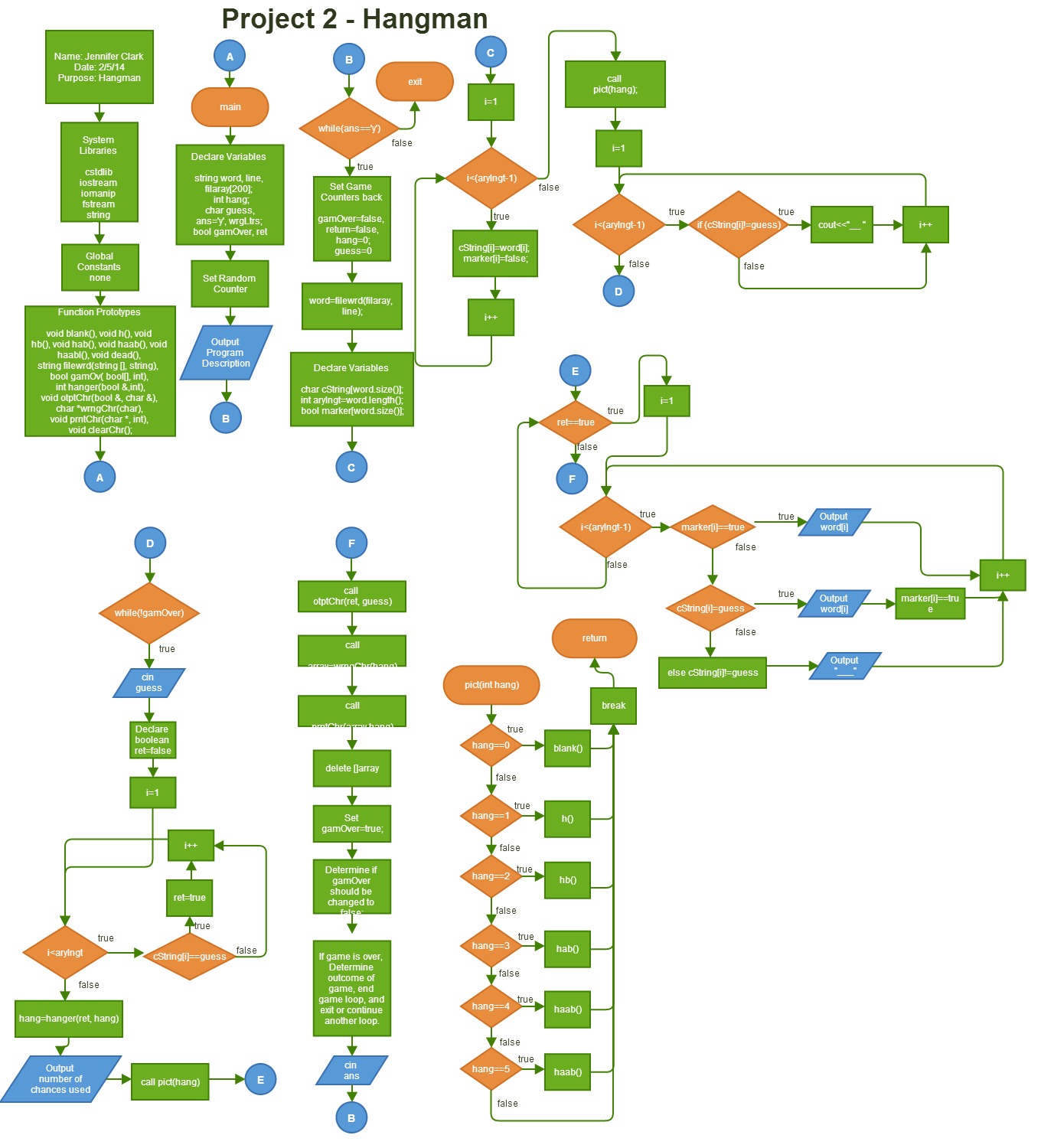
IDE: Netbeans

This is a more complex version of the first Hangman game from Project 1. For the user, not much has changed other than some cleanup of the interface. One addition was the output of the array of incorrect letter-guesses, making it easier to see which letters have already been guessed incorrectly.

The most difficult changes to make were behind the scenes in the code. In order for the incorrect guesses to be output, I had to write each wrong guess into a file in one function. I created another function to read in the characters from that file and put them into a dynamic array (the array grew with each incorrect guess). Then I used another function to print out the array to the user.

I still see room for improvement. It would improve program readability to put more of the code into separate functions. I also wanted to keep a running total of the number of wins and losses of a player in one sitting, but outputting the incorrect-guess array was more difficult to code than I thought it would be and I didn’t get that far in the allotted time.

**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’)*

*Increment the integer by one*

*Call pict(hang)*

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

*Output an underscore mark to signify an un-guessed character*

*Increment the integer by one*

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

*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”*

*Set hang equal to the hanger(ret, hang) function.*

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

*Call pict(hang)*

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

*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*

*Increment the integer by one*

*Call the function otptChr(ret, guess) to write the next wrong-letter to the file, if necessary.*

*Fill the char array and cout wrong\_guess\_output.dat*

*Output the array of wrong character guesses from the prntChr(array,hang) function.*

*Delete the character array from the program.*

*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 |

**Major Constructs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Name** | **Location** | **Description** |
| while loop | while (ans=='y') | Line 138 - main | Loops the game while the player still wants to play |
| if loop | if (cString[i]==guess)ret=true | Line 85 - main | Determines if the guess is correct or incorrect for a character in the word |
| for loop | for (int i=0; i<arylngt; i++) | Line 84 - main | Allows the above if-loop to check for each character in the word |
| if/else if loop | if (marker[i]==true); else if (cString[i]==guess); else if (cString[i]!=guess) | Line 93-100 - main | Outputs the word and uses blank spaces if a character has not yet been guessed |
| do while | do{}while(!gamOver) | Line 77-122 | Loops the guessing game |
| switch | switch(hang) | Line 202-221 - pict(int hang) | Determines what graphic should be output |
| pass by value | hanger(...int hang) | 191-199 - hanger(bool &ret,int hang) | Increments the wrong-guess counter "hang" |
| pass by reference | hanger(bool &ret...) | 191-199 - hanger(bool &ret,int hang) | Checks the boolean "ret" to determine if an incrementation should occur |
| dynamic array | array | Line 158-174 - \*wrngChr(int hang) | Creates an array of characters based on the number of characters in a file |
| passing arrays between functions | \*wrngChr(int hang) & prntChr(char \*a,int hang) | Line 151-174 | The first function creates the array and passes it in, the second function prints the array |
| returning data types through a function | string filewrd(string filaray[], string line) | Line 222-241 | Returns a String datatype to main |
| read from a file | \*wrngChr(int hang) | 158-177 | Reads in a list of characters from a file |
| write to a file | otptChr(bool &ret, char &guess) | 179-189 | Writes a character to a file if the guess is wrong |
|  |  |  |  |

**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>