Assignment 2 – Hangman Report Template

Catie Bronte

CSE 13S – Winter 24

Purpose

This is a basic game of hangman, but the user is trying to guess phrases over a word. The program first prints some small gallow art, followed by an underscored line where correct guessed words will pop up. Below that is a list of eliminates letter, and the final line at the very line prompts the user to guess a letter. With each guess the screen updates with either an added limb on the gallow art with the guessed letter appearing alphabetically on the eliminated line if the letter is incorrect, or, the same gallow art with the letter appearing on the phrase line as it is located in the phrase if it is correct. This keeps happening untill the user reaches their max incorrect guesses or they win. On a win or loss a respective statement is printed, and the program returns. (returns 0 or 1?)

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question. To fill in the answers and edit this file, you can upload the provided zip file to overleaf by pressing [New project] and then [Upload project].

Guesses

One of the most common questions in the past has been the best way to keep track of which letters have already been guessed. To help you out with this, here are some questions (that you must answer) that may lead you to the best solution.

- How many valid single character guesses are there? What are they?

  The player can guess any lower case letter in the English alphabet, which comes out to 27 max unique guesses.

- Do we need to keep track of how many times something is guessed? Do you need to keep track of the order in which the user makes guesses?

  We fortunately do not need to keep track of how many times a character is guessed, or which order they make guesses. Hangman does not depend on these factors.

- What data type can we use to keep track of guesses? Keep in mind your answer to the previous questions. Make sure the data type you chose is easily printable in alphabetical order. [1]

  We can use an array of characters to keep track of the guesses, since you can derive a uniform alphabetical order from each character.

- Based on your previous response, how can we check if a letter has already been guessed. [2]

  To check if a letter has already been guessed, there will be a for loop to iterate over a guessed characters array, and will check each item against the guessed character.

[1] Your answer should not involve rearranging the old guesses when a new one is made.

[2] The answer to this should be 1-2 lines of code. Keep it simple. If you struggle to do this, investigate different solutions to the previous questions that make this one easier.

Strings and characters

- Python has the functions `chr()` and `ord()`. Describe what these functions do. If you are not already familiar with these functions, do some research into them.

  `chr()` takes an integer value between 0 and 1,114,111 , and returns its respective unicode character. For example `chr(97)` returns 'a'. `ord()` is the exact opposite of `chr()`. It takes a Unicode character and returns its Unicode representative integer. For example `ord('a')` returns 97.

- Below is some python code. Finish the C code below so it has the same effect. [3]

  ```
  x = 'q'
  print(ord(x))
  ```

  C Code:

  ```
  char x = 'q';
  printf("%d", x);
  ```

- Without using `ctype.h` or **any** numeric values, write C code for `is_uppercase_letter()`. It should return false if the parameter is not uppercase, and true if it is.

  ```
  #include <stdbool.h>
  char is_uppercase_letter(char x){
      if ( 'x' > 40 && 'x' < 91) {
      return true;
      }
      else {
      return false;
      }
  }
  ```

- What is a string in C? Based on that definition, give an example of something that you would assume is a string that C will not allow.

  A string In C is an array of characters that end in a null terminator. For example, if you had an array of "abc" vs "abc backslash zero", C will throw a hissie fit over the first one. :)

- What does it mean for a string to be null terminated? Are strings null terminated by default?

  When the compiler or loop or what have you is iterating through a string, it will continue doing so until it hits the null terminator. The null terminator exists to signify the end of the string. By default, strings are indeed null terminated.

- What happens when a program that is looking for a null terminator and does not find it.

  If a program looks for a null terminator that is not there, it will keep looking forever.

- In this assignment, you are given a macro called `CLEAR_SCREEN`. What is it's data type? How and when do you use it?

  The macro is of type string since it is declared with "" and not ''. Essentially we print it with printf at the top of a round. Basically whenever the screen changes.

Testing

There are 2 testing elements to this program. test helpers.c and test functionality.sh. test helpers tests the 4 main functions to make sure they are functioning properly. Test functionality does the same but for the entire main program, by comparing the differences of the outputs.

[3] Do not hard code any numeric values.

How to Use the Program

The executing of this function is pretty simple. First type ./hangman "hangman phrase you want to play with" make sure this phrase is either 1 word or in quotations. The only special characters you can use are - ' ' or '. From there, input one letter guesses until you win or lose.

Program Design

For the main hangman program, it is broken into 2 main parts. The initial declaration of all of the variables, and the actual game loop. From there, look for comments to describe the minor operations of the program.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

References