Assignment 2 – Hangman Report Template

Debi Majumdar

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

In this assignment, we focus on the fundamental elements required for creating the Hangman word game in C. The game revolves around representing words using characters in C, where each character is essentially a signed 8-bit integer according to the ASCII table. We delve into the concept that a string is essentially an array of characters. Despite the potentially morbid themes associated with Hangman, it is highlighted that the game can serve as an effective tool for teaching vocabulary.

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?

  There are 26 valid single character guesses, representing each letter of the alphabet.

- 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?

  I do not think we need to keep track of how many times something is guessed, nor do we need to keep track of the order in which the user makes guesses. Only the information about whether a letter has been guessed or not is relevant.

- 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 a boolean array of size 26 to keep track of guessed letters. Each index corresponds to a letter, and a value of true indicates that the letter has been guessed.

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

- 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, we can use the boolean array. For example, if guessedLetters is the boolean array and letter is the current guess we can check it.

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(): This function in Python takes an ASCII value (an integer) and returns the corresponding character. ord(): This function in Python takes a character and returns its ASCII value

- 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", (int)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){
  bool is_uppercase_letter(char x) {
     return x >= 'A' && x <= 'Z';
  }
  ```

- 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.

  In C, a string is an array of characters terminated by a null character. However for example, something assumed as a string but not allowed in C could be an array of characters without a null terminator.

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

  A string in C is null terminated when it is ended with a null character. Strings are not null-terminated by default. You need to explicitly include the null character at the end.

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

  If a program is searching for a null terminator and does not find it, it may result in reading beyond the allocated memory, leading to undefined behavior and potentially crashing the program or causing unexpected results.

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

  Macros in C are symbolic constants, and the data type is a string, which can be used whenever the user wants to clear the screen.

[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.

[3] Do not hard code any numeric values.

Testing

List what you will do to test your code. Make sure this is comprehensive. [4] Remember that you will not be getting a reference binary [5].

The comprehensive testing plan for the Hangman game involves thorough verification at various levels. Unit testing will assess individual functions, ensuring they handle different inputs and produce expected outputs. Integration testing will examine the overall game flow, including user input processing and state updates. Boundary testing will validate the game's capability to handle diverse secret phrases, special characters, and correct display of game elements. Testing will cover winning and losing scenarios, command line argument parsing, proper screen clearing, and display functionality. Memory leak and error testing, code coverage analysis, and interactive playthroughs will uncover potential issues, while compatibility testing ensures cross-platform functionality. Finally, performance testing will assess the game's efficiency under stress conditions.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

Using the Hangman game is straightforward. Begin by compiling the program with the command `gcc -o hangman hangman.c hangman_helpers.c -std=c99`. Then, run the game with `./hangman "your secret phrase"`, replacing the placeholder with your desired phrase enclosed in double quotes. Guess one letter at a time during the game, and the gallows, phrase, and eliminated letters will be updated accordingly. You win by guessing the entire phrase correctly, and you lose if you exceed the allowed mistakes. For optional visual enhancements, compile with the flag `-D CLEAR_SCREEN`. If you encounter issues, refer to the documentation or seek support. Enjoy playing Hangman!

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[6]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

Maintaining the Hangman game involves working with several key data structures and algorithms. The program extensively uses arrays for representing the gallows, secret phrases, and eliminated letters, while

[4] This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[5] The output of your binary is not the only thing you should be testing!

[6] This is my footnote.

boolean arrays track guessed letters. Macros, like `CLEAR_SCREEN`, and constants, such as `LOSING_MISTAKE`, enhance visual features and define game parameters. The code may also employ structures to organize the game state. Algorithms embedded within functions handle guessing logic, input validation, and core game operations, determining winning, losing, and updating the game state based on player input. To facilitate future maintenance, we can try to ensure a modular and well-documented code structure, with clear separation of concerns and adherence to industry coding standards. Thorough documentation, including comments explaining function purposes and complex logic, is essential for easy comprehension and debugging. By following these guidelines, maintaining and enhancing the Hangman game will be a more straightforward and organized process.

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.

The Hangman game code adopts a top-down modular structure, with distinct functions serving specific purposes to manage the game state, handle user input, validate guesses, and display the current status. The initial function initializes the game, sets up the gallows, and prompts the user for the secret phrase. It checks adherence to specific criteria, including character types and length. Core game logic, such as checking guessed letters, reading user inputs, and validating lowercase letters, is implemented in separate functions. The code also features functions for updating the game state based on user guesses, determining winning and losing conditions, and managing the visual display of the game on the screen. The implementation emphasizes modular design, adherence to coding standards, and clear separation of concerns. Each function is designed with a specific role, fostering readability, maintainability, and extensibility. Extensive comments and documentation accompany the code to facilitate future understanding and modifications.

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.**

This C program implements a simple hangman game. Let's break down the main functions:

`bool all_guessed(const char *secret, const char *guessed)`: This function checks whether all the letters in the secret phrase have been guessed. It iterates through each character in the secret phrase, excluding punctuation, and checks if it has been guessed.

`int compare_char(const void *a, const void *b)`: This function is a comparison function used for sorting characters. It's used in the `qsort` function to sort eliminated letters alphabetically.

`void print_game(const char *arts[], int gallows_state, const char *secret, const char *guessed_letters, const char *eliminated_letters)`: This function prints the current state of the hangman game. It displays the hangman ASCII art corresponding to the `gallows_state`, prints the phrase with guessed letters revealed, and displays the eliminated letters in alphabetical order.

`int main(int argc, char *argv[])`: The main function orchestrates the hangman game. It takes a secret word or phrase as a command-line argument, validates it, and then enters a loop where the user guesses letters. The game state is continuously printed, and the loop continues until the user either guesses the entire phrase or exhausts the allowed number of incorrect guesses.

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.