Hangman

Adam Wu

2023 December 21

1 Purpose

Hangman is a classic word game that most people know. Its origin came from Europe during the 17th and 18th century. Criminals who were going to be sentenced to death could demand for a 'Rite of Words and Life' [1]. For this project, the goal is to practice writing code in C++ (specifically playing around with string arrays). Thus, the best way to do that is to create hangman in the terminal!

2 How to Use the Program

First clone the repository to get all the source files. The next step is to go to folder where this project is on your local machine and go to the *src* folder. To run hangman you will need to compile the program first, then you can run it. In the next section, you will find out how to compile, run, and removal of the program.

2.1 Compiling the program

You will just need to type *[*make] and the program will start compiling. If you have never programmed before, a demonstration of this program compiling will be shown below.

```
$ make
c++    -c -o hangman.o hangman.cpp
c++    -c -o hangman_helpers.o hangman_helpers.cpp
g++ -std=c++17 -Wall -g -o hangman hangman.o hangman_helpers.o
$
```

after successfully compiling, try and type **ls**. When you do, you will notice a new file with a different color have emerged. That is the program executable. With that, we can move on to learning how to run the program.

2.2 Running the program

Now to run the program, you will need to do the following:

```
$ ./hangman <secret word or phrase>
```

For this program, the allowed characters are lower cased characters and punctuation like a space, an apostrophe, and a dash. Other punctuation and capital letters will be unhallowed. From there on, the game should be very intuitive.

2.3 Removal of the program

To remove the executable, you will need to type *make clean*. Soon after, you can type *ls* to check that your folder does not contain the colored executable anymore.

3 Program Design

This program only consists of 3 files. hangman.cpp, hangman_helpers.cpp, and hangman_helpers.h. The hangman helper file help with clearing the screen, the ascii art of the hangman, and validating the string array. The main hangman file then utilizes the helper function for the main game.

In the beginning of the game, it will clear the screen and display the gallows along with a line of the phrase and a line of the eliminated characters. It will also prompt a new line to have the player guess a letter. The following will be an example and the phrase will be: **we'll see**.

```
 -----
 |     |
       |
       |
       |\
       | \
     __|__\


    Phrase: __'__ ___
Eliminated:

Guess a letter:
```

If you look closely at the example, you will notice that the 'phrase' and 'eliminated' are aligned properly. As player enters the correct letters, the program should also update. A new gallows will be printed, the phrase and eliminated will also be updated. If the player were to lose (getting 6 incorrect letters) then the output should be the following.

example losing screen

```
 -----
 |     |
(_)    |
/|\    |
 |     |\
/ \    | \
     __|__\


    Phrase: w_'ll ___
Eliminated: abnopv

You lose! The secret phrase was: we'll see
```

if the player were to win, then the only difference from the losing screen is that the message is **You win! The secrect phrase was: we'll see** in our case.

Furthermore, inside the src folder, there also consists of a hangman_helperTest.cpp to help test the hangman helper functions. It is not necessary for you to use it, it is good for me to test if my functions worked properly.

3.1 Data Structures

This program only required the usage of an array that is static memory (heap memory was unnecessary for this project). The array was used to store the ascii art of the gallow.

3.2 Algorithms

In the hangman game, I made it so that the eliminated words would be sorted in alphabetical order. To do the sorting, I used a linear sorting algorithm (one that is similar to insertion sort). Below will be the psuedo-code for the way I have done it.

update(eliminated)

```
update(eliminated):
    if eliminated.empty():
        eliminated += read
    else:
        int i = 0;
        while (i < eliminated.length() && readChar > eliminated[i]) {
            i++;
        }
    mistakes++;
```

3.3 Function Descriptions

Inside the hangman helpers header file we will have the following functions.

bool string_contains_character(const std::string& s, char c);

This function will check if a string contains a character c. If it does, it will return true. False otherwise.

char read_letter();

This function will read a letter from stdin and return the read character back.

bool validate_secret(const std::string& secret);

This function will take in a string and will validate the string (aka the secret). The string is valid as long as it contains all lowercase letters or if the string contains punctuation like a space, an apostrophe, or a hyphens. If the secret (string) is valid, then it will return true. False otherwise.

bool is_lowercase_letter(char c);

This function will check if a character c is lowercase. If it is, then it will return true. False otherwise.

4 Results

The hangman game works like the usual hangman game where the users may enter a secret phrase and guess the characters. If they guess 6 incorrect characters, then they lose. Additionally, the game works fine with no leaks or errors.

4.1 Error Handling

This program checks if the number arguments is correct. If it is correct, then the program will continue. Otherwise, an error message will appear to show you how to use the program.

References

[1] Maximus Florence. From history to history class: the origin of classroom games. https://chschipper.com/2019/12/from-history-to-history-class-the-origin-of-classroom-games/, 2019. Accessed: 2023-12-21.