Nicholas Harrell – VP66491

Nick Allgood

CMSC 341

11 December 2022

Hangman Writeup

To play hangman, run the driver / make val. To test the hangman functions, run mytest / make valtest. No other data structures were used in this program, only tries and nodes.

Hangman.cpp

- Holds the trie data structure and the game functions
- Trie functions
  - Hangman()
    - Constructor for the trie. There is no progress in the game and no words in the trie
  - ~Hangman()
    - Deconstructor for the trie. Calls delete m_root, where m_root is a node with a recursive destructor
  - validString(string s)
    - checks if the string consists of only alpha characters. Returns false otherwise.
  - Insert(string s)
    - Adds a valid word to the trie. For the length of the string check if each character exists in consecutive nodes. If it doesn't create a new node to hold the character. The index is determined by s[i] – 'a', giving an index of 0 to 25. The node's end value is then set to true to show that a valid word ends there.
  - Search(string s)
    - Similar to insert, except it won't create a new node if the character doesn't exist in the vector and will instead return false. Otherwise, if it reaches the end of the word, return true.
- Game functions
  - isInString(string s, char c)
    - check if a character is in a string. Helpful for determining if a guess was correct
  - display()
    - draws a picture of the current hangman, then shows the user progress and what they have guessed already
  - getWrong() & incWrong()
    - return the number of wrong guesses; increment the number of wrong guesses
  - loadWords(string data)
    - data is the name of a csv file. Loads each word into both the trie and a vector. If no words are loaded, the function returns false to signify the game should not continue

- pickWord()
  - randomly picks a word to use from the vector of stored words and verifies that it is in the trie. Then edits m_secret to display the write amount of spaces
- getInput()
  - gets the user input and converts it to lowercase. Then only returns the first character and adds it to the m_guessed vector.
- addToGuessed(char c)
  - takes in a char c andchecks if it is already in the m_guessed vector. If it isn't, add it to the vector
- displayGuessed()
  - prints out all of the characters in m_guessed
- playGame()
  - picks a random word and sets it as the secret word. Then runs a while loop until the user runs our of guesses or solves the word. If the user's input is in the secret string, edit m_progress to display where it is. Otherwise, increment wrong guesses. If the user's progress equals the secret word, then they have won and will exit the while loop. Then the game values are reset and prompts the user if they want to play again, returning the appropriate boolean.
- Reset()
  - Restores all game values back to their original state (m_wrong, m_secret, m_progress, m_guessed)

Hangman.h

- Holds the node object and defines the hangman class
- Node
  - Node(char l){
    - Sets the nodes letter to hold l, end value to false, and a vector that holds 26 null Nodes.
  - ~Node()
    - Recursive deconstructor for nodes. For each node in next, delete it. After that is done, clear the next vector.

Driver.cpp

- Creates a trie, loads data, and repeatedly plays the game until the user says no

Makefile

- driver
  - makes the driver executable
- mytest
  - makes the mytest executable
- hangman.o
  - used for the executables
- val
  - makes driver and then runs valgrind on it

- valtest
    - makes mytest and then runs valgrind on it

food.txt

- csv file that holds all of the strings used in this hangman game.

Mytest.cpp

- only tests the trie functions of hangman, not the game functions
- Tester
    - Construct()
        - If the default values are not what they are supposed to be, exit
    - insertNorm()
        - normal case insertion to the trie. If a route cannot be traced that spells out the inserted word, return false
    - insertError()
        - error case, try to insert something that is invalid (not made of only letters). Should not insert
    - searchNorm()
        - normal case search for the trie. If the search function doesn’t find the inserted word, return false
    - searchError()
        - error case search for the trie. Should not find a string that was not insert into the trie