Team 1 Project 3B

PREPARED BY

Joe Simon

Eric Vaughn

Jordan Pham

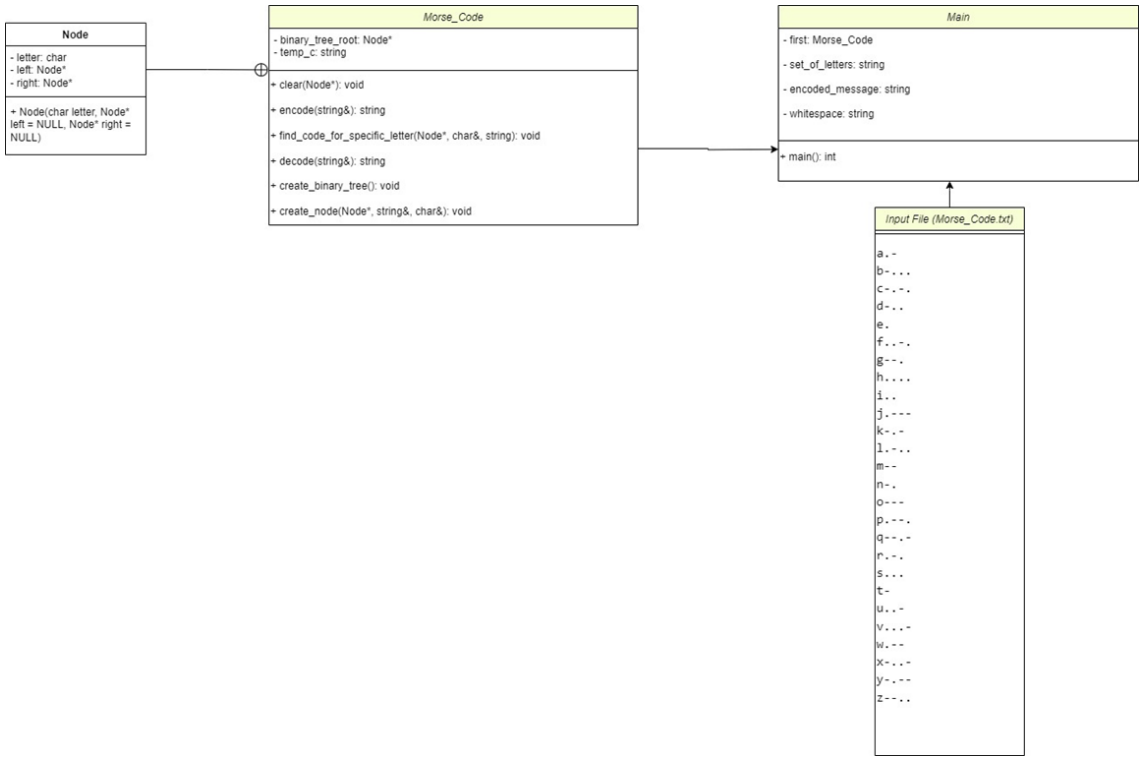
2) System Design

Our system has two .cpp files—Main.cpp and Morse_Code.cpp—and one header file, Morse_Code.h. The Main file contains the int main() function which creates a Morse_Code object first. The user is then asked to enter a series of letters to be encoded and decoded by Morse code. These letters are stored within the string set_of_letters. The encoded_message string is then initialized by the firsts' encode function of the series of letters. The original message and the encoded_message are then output to the user. The encoded_message is then decoded under firsts' decode function, which is also output to the user.

After the Morse_Code object is created in the main function, the create_binary_tree() function is called within the default constructor. This function reads from the Morse_Code.txt file, which contains the Morse codes for each lowercase letter. Within a while loop, it calls the create_node() function, which adds a node, containing a letter from the .txt file, to the binary_tree_root. Once the encode function is called, it goes through each letter within the parameter string letters and passes the letter into the find_code_for_specific_letter() function. This function recursively finds the letter and then assigns the string temp_c variable with the code. In the encode function, the temp_c variable is added to a string of codes, which is then returned. When the decode function is called, it also recursively finds the letter in the binary tree, adds the letter to a string of letters, and then returns those letters.

There are several data structures used in our system. First, we used the string data structure for storing several variables in the Main program and the Morse_Code files such as letters, codes, decoded_message, etc. Second, we used the iterator data structure for looping through strings. Third, we used a binary tree to create the Morse code tree with the node binary_tree_root as its root from the text file Morse_Code.txt.

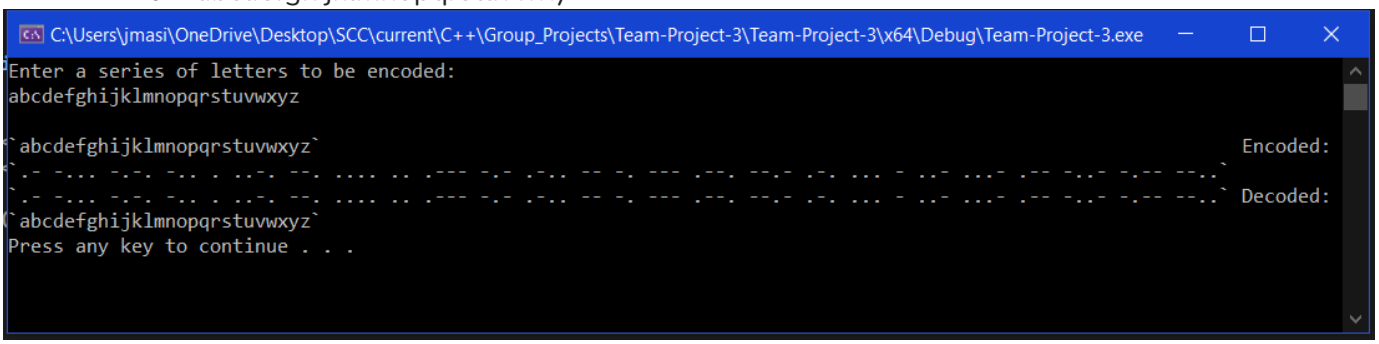
3) UML Class Diagram



4) Two Test Cases

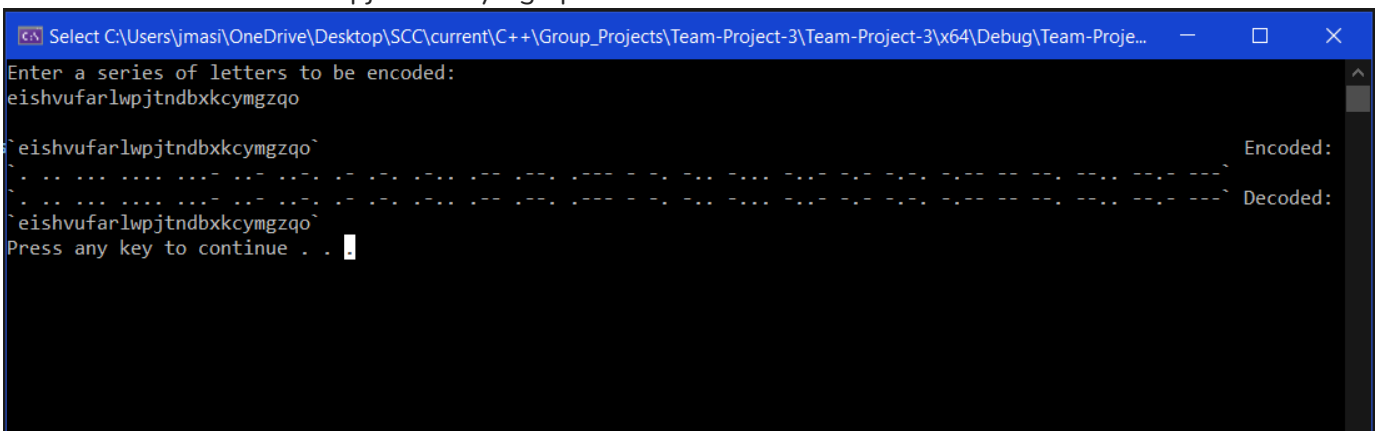
Test Case 1:

- Expected output when encoded:
  - .- -... -.-. -.. . ..-. --. ....--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .-- -.- -.-- -- ..
- Actual output when encoded:
  - .- -... -.-. -.. . ..-. --. ....--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .-- -.- -.-- -- ..
- Expected output when decoded:
  - abcdefghijklmnopqrstuvwxyz
- Actual output when decoded:
  - abcdefghijklmnopqrstuvwxyz



Test Case 2:

- Expected output when encoded:
  - `. .. ... .... ...- ..- ..-. .- .-. .-.. .-- .--. .--- - -. -.. -... -..- -.- -.-. -.-- -- --. --.. --.- --`
- Actual output when encoded:
  - `. .. ... .... ...- ..- ..-. .- .-. .-.. .-- .--. .--- - -. -.. -... -..- -.- -.-. -.-- -- --. --.. --.- ---`
- Expected output when decoded:
  - eishvufarlwpjtndbxkcymgzqo
- Actual output when decoded:
  - eishvufarlwpjtndbxkcymgzqo



For both the encoded function and the decoded function, the expected and actual outputs are the same (besides the formatting error in test case 2, though this is a Microsoft Word problem and not a code problem).

5) Contributions

1. Joe
    a. Main system design: Was the main contributor towards the overall design of the program including things like its header files, functions, and main file.
    b. Attended team meetings: Attended every team meeting and discussion to help brainstorm ideas.
    c. Debugging: Contributed to the team overall to help identify and fix bugs.
    d. Report and Class Diagram: Created the UML class diagram as well as the project report.
2. Eric
    a. Programming: Contributed ideas and programming skills to help with the design of the program
    b. Attended team meetings: Attended every team meeting and discussion to help brainstorm ideas.
    c. Debugging: Contributed to the team overall to help identify and fix bugs.
3. Jordan
    a. Programming: Contributed ideas and programming skills to match the layout of the program determined by Joe.
    b. Logic: Contributed many logic-based corrections and ideas.
    c. Attended team meetings: Attended every team meeting and discussion to help brainstorm ideas.
    d. Debugging: Contributed to the team overall to help identify and fix bugs.

6) Improvements

1. Include functionality for more symbols and capital letters.
2. More team project meetings
    - Team meetings were effective, but very sparce. Progress could've been made a lot faster had we communicated more.
3. In-line comments:
    - More comments to organize thoughts would've helped in communicating ideas between teammates as well as our future userbase.
4. Improve the efficiency of the `encode` function:
    - Currently, the time complexity of the encode function is O(n * h) where n is the size of the input text, and h being the height of the tree. This is very slow, so optimizing the algorithm more would make it easier to maintain and handle larger and more complex sets of user input.
5. Make the main function more user-friendly (GUI interface)