Morse Code Program Developer Manual

Program Functions:

1- void textMorse()

This Functions has not any parameters but is asked the user to enter the required data. The function scan a text from the user to convert in it into morse code. It has a for loop that keep comparing every letter of the pre-entered text with the struct array that contains every letter and its equivalent in the morse code. The main for loop ends when the text is over. It has also a for loop to count the dots a dashes in the code to print the statistic after the converted text.

2- void MorseText (Node *tree)

This function has one parameter which is the binary search tree that has the data of the letters and its representations in morse code. It asks the user to add an input of morse code type, is has one more pointer (temp) that points to the head of the tree to move freely during the search without affecting the balance of the tree.

The program checks the length of the scanned morse code and then it processes it through a nested loop that repeated (x) times where (x) is the length of the text.

The temp pointer moves to the left when it is dot and to the right when it is dash. The function prints the data of the node that that temp points to when there is a space, that means a space between two letters. It prints space when morse code contains a " / ". The function also counts the number of English letters that converted text include. Then

when the function finds a space the temp is back to points to the head of the tree again that's why I do not use the head itself since It will not have a reference to balance the tree again.

3- void FILE_MorseText(Node *tree)

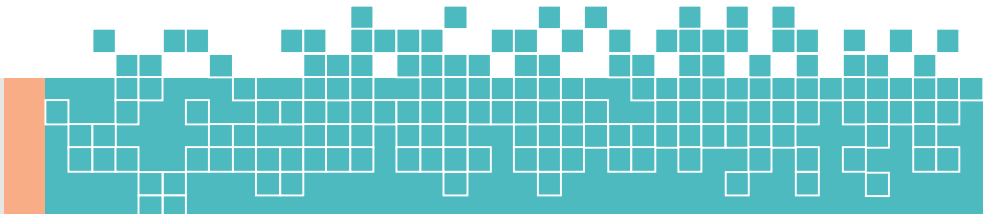
It follow the same functionality as MorseText(Node *tree) but the functions gets its input from the file named "Morse.txt" and prints its output in the file "Text.txt". It has a for loop that scan line by line and process it to print the output in "Text.txt" file.

4- void FILE_textMorse()

It follow the same functionality as textMorse()
but the functions gets its input from the file named "Text.txt" and prints its output in the file "Morse.txt". It has a for loop that scan line by line and process it to print the output in "Morse.txt" file.

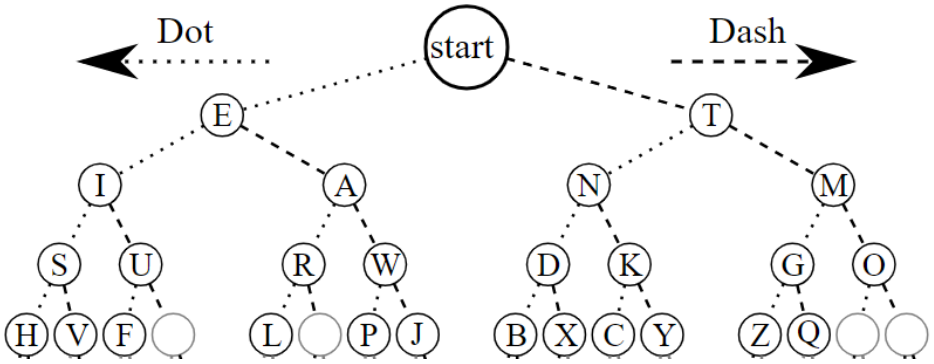
5- Node* create()

The function creates a node that has data and 2 pointer nodes that points to the left and right. The left always refers to a dot while the right refers to a dash.



6- Node* Insert(Node *tree, char letter, char *morse)

The function inserts the English letters in the binary tree according to a process. It uses the struct array in the insertion. It uses the morse representation, if it is dot goes left and if it is dash goes right. Once the morse representation is over, it inserts the letter where the pointer refers. Kindly, find the Morse Code Diagram attached below.



7- void delete_Tree(Node *tree)

This Function gets the pre-inserted binary tree as a parameter to delete every element in the tree which not equal NULL step by step. Once the tree equal NULL the recursion will stop and in this way the tree will be deleted.

8- int main()

The main functions prints all the valid option for the user for 6 cases.

1. It calls the textMorse function that converts from text to Morse

2. It creates the start node which is empty and them it has 2 NULL pointers, then it calls insert functions that fill the binary tree with the letters according to the dashes and dots, it calls the function MorseText() that converts from Morse to Text.

3. It do the same as Number 2 converting from Morse to Text but using files as an input and printing the output in "Text.txt".

4. It follows the same way as case number 1 but it converts from text to morse and uses file handling. Scanning the input from "Text.txt" to print the output in "Morse.txt".

5. It prints "Invalid Input " message in case the user entered an incorrect option.

Data structures and Design Considerations:

Structures:

1. Node Struct
   This struct is used in creating the nodes of the binary tree and it 3 elements. Data, left pointer, and right pointer.
2. Convert Struct
   This struct is used to create an array that contains the English letters and their representations in morse code.

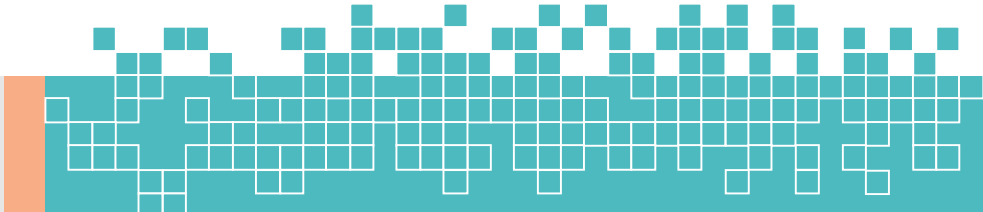
Binary Tree:

1. Node *tree

   The code contains only one binary tree. The tree has many nodes and every node contains a data and 2 pointers that refers to 2 more nodes that have data and every node of them has to more pointers. In the binary tree the node must contain 2 children at most.
   The tree is considered as the most suitable data structures to store the Latin letters to get their morse code. It is very fast and accurate.

File Format:

1. "Morse.txt"
   This file is used for 2 cases:
   1. to print in the output while converting from text to morse using the file handling.
   2. As well as it's also used to scan the input while converting from morse to text.



2. “Text.txt”

   On the other side this file is used in 2 other case:

   1. To print the output while converting from Morse to Text using the file handling
   2. To scan the input while converting from text to morse using the file handling.

Requirements to build the program:

- There are not any external libraries required to build it.
- The program may be built easily using the standard c compiler.
- The program does not require and special hardware.