-------------------------------------------------------------------------------------------------------------------------------

Morse Code Project

Due: April 30, 2019

Creators:

Connor Kasarda

Tiago Araujo

Matt Sharp

Bailie Allemand

Jeffrey Michaelis

-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------

Project 2: Morse Code

UW COSC 2030, Spring 2019

Goal: Create program for translation of Morse Code to English, and English to Morse Code

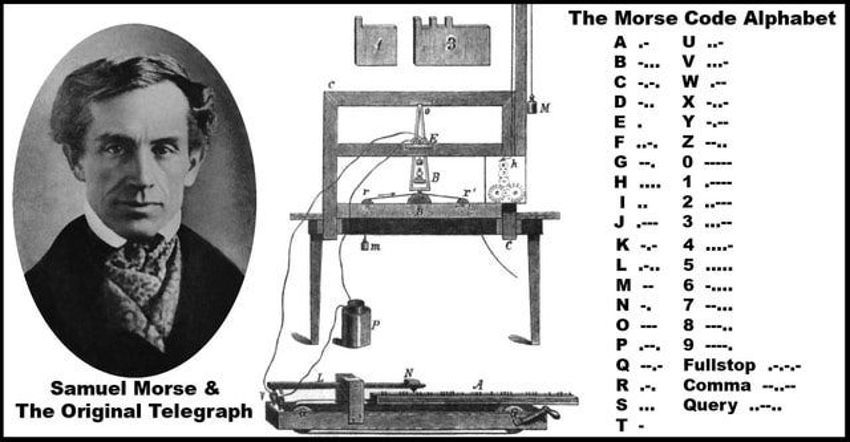
We chose to create both a C++ and Python program to carry out the project requirements. Sections 1 through 4 of this proposal include the ADT we implemented, references to the data items we submitted, operations(pseudocode) for both programs, and an example and description of each program in action.

-------------------------------------------------------------------------------------------------------------------------------

History of Morse Code:

In the United States in the 1830s what is known today as Morse code was created by Samuel Morse and Alfred Vail as a form of long distance communication. The code was used to transmit messages through the use of telegraph wires. It wasn’t until 1844 that the first message in Morse code was sent.

The code itself was comprised of sequences of dots and dashes, or short marks and long marks. There was a different sequence assigned to each letter of the English alphabet and to numbers.

Originally, when the code was transmitted it was rendered as marks on a piece of paper and then translated by the operators. However, to improve efficiency as the operators learned the code they were able to exclude the paper process and replace it with a receiver that created short and long beeps that the operators could translate.

-------------------------------------------------------------------------------------------------------------------------------

Section 1

Abstract Data Type

-------------------------------------------------------------------------------------------------------------------------------

For translating the Morse code file into English, we constructed a binary tree. The following sections go into greater detail of how the binary tree is implemented.

-------------------------------------------------------------------------------------------------------------------------------

Section 2

Data Items

-------------------------------------------------------------------------------------------------------------------------------

Binary tree

Message

Character table

-------------------------------------------------------------------------------------------------------------------------------

Section 3

Operations

-------------------------------------------------------------------------------------------------------------------------------

**C++(Pseudocode):**

This program translates messages from English to Morse code, as well as from Morse code to English.

Structure Node creates a node that contains a character value and has a left and right child node; both defaulted to NULL to be used to create the binary tree used for the Morse to English translation.

Function addLetter(character variable, string variable, Node variable)

Function adds letters and characters to the binary tree that is used for decoding from Morse code to English.

If string variable size is equal to 0

Then place char variable as the value for the current Node

Else if a (‘.’) is encounter at the current position in the string variable

Moves to the current Nodes left child (default starts at the root of the tree)

If the current node does not have a left child

Create a left child for the node

Recursively calls function by removing first character from string variable to keep running until the end of the string variable size.

Else if a (‘-’) is encounter at the current position in the string variable

Moves to the current Nodes right child (default starts at root of tree)

If the current node does not have a right child

Create a right child for the node

Recursively calls function by removing first character from string variable to keep running until the end of the string variable size.

End

Function decodeTreeConstructor(string variable, Node variable)

Function constructs the Binary tree used for the Morse to English translation in this program.

` Opens file used to construct tree.

If the file is open

Invariant: letter always has first letter of line, or letter to be represented, and morse has rest of same line from position 2, the morse string equivalent

While there is still a line to read in the file

The character to be inserted into a specific node will be assigned to a char variable

The string of dots and dashes will be assigned to a string variable

Will call the addLetter Function in order to insert the character in correct position in the binary tree. Function will hold the values of the character, string, and the root node as its variables.

Closes the file used to construct the tree.

End

Function findLetter(string variable, Node Variable)

Function finds a specific character represented by the Morse code string that is given

If the size of the Morse code string is 0

The character at the current Node is the letter represented by the Morse code string.

Prints out the letter found.

Else if a (‘.’) is encountered at the current position in the string variable

If a left child does not exist from the current node

Prints “Error” message to tell user that a letter was not found for that Morse code string

Else Recursively calls function by removing first character from string variable to keep running until the end of the string variable size, and by moving to the right child of the current node.

Else if a (‘-’) is encountered at the current position in the string variable

If a right child does not exist from the current node

Print “Error” message to tell user that a letter was not found for that Morse code string

Else Recursively calls function by removing first character from string variable to keep running until the end of the string variable size, and by moving to the right child of the current node.

End

Function E2M(string variable1, string variable2)

Function translates a message from English into Morse code.

Create a string variable that will hold values of the dots and dashes that are given to a specific character

Open a file whose name is the value of string variable2, which should contain the message that is expected to be translated

Find the length of the message in the file and stores this value to the variable to be used later.

Create a buffer for the English message text

Read in the complete message in the file using the file length to ensure full message is read in

Open a file whose name is the value of string variable1, which should contain a table that represents the characters and how they translate into Morse code.

Find the length of the file that contains the table and stores this value to a variable for later use

Create a buffer for the Morse table text

Read in the complete Morse code table in the file using the file length to ensure that the full table is read

Invariant: i is always less than character length of text in message file

incrementing by 1 each iteration

Invariant: j is always less than the character length of the text in Morse table file

current character in the message is found in the table, incrementing by 1 each iteration

If the current character from the translation file is found in the Morse table file

Make an int variable equal to 2, to skip the character that is given in the Morse code Table

Invariant: morse\_string is either part of or the entirety of the morse string equivalent of the letter being translated

While the Morse code tables buffer is either a (‘.’) or a (‘-‘)

If the current character in the table is a (‘.’)

Add a (‘.’) to the string holding the current characters dots and dashes

Else if the current character in the table is a (‘-‘)

Add a (‘-‘) to the string holding the current characters dots and dashes

Increment the int variable by 1 so the while loop can read in the next dot or dash (as long as there is one)

Print out the string holding the dots and dashes for the specific letter

Reset the string holding the dots and dashes back to an empty string

Break from the if

Else if there is a space next in the translation file

Print 3 spaces for clarity purposes

Reset the string holding the dots and dashes back to an empty string

Break from the ElseIf

Reset the string holding the dots and dashes back to an empty string.

Close the file holding the message to be translated.

Close the file holding the Morse Code Table

End

Function M2E(Node variable, string variable)

Function translates a message from Morse Code into English using a binary tree.

Create a string variable that will hold characters for the message being translated

Open a file whose name is the value of string variable, which should contain the message that is expected to be translated

Find the length of the message in the file and stores this value to the variable to be used later.

Create a buffer for the Morse code message text

Read in the complete message in the file using the file length to ensure full message is read in

Invariant: Morse is always either part of or the entirety of the morse code string equivalent to be decoded

incrementing by 1 each iteration

If the current character is a (‘.’)

Add a (‘.’) to the string holding the dots and dashes for the current character to be decoded

ElseIf the current character is a (‘-’)

Add a (‘-’) to the string holding the dots and dashes for the current character to be decoded

Else if the next 3 characters are empty spaces, indicating the start of a new word

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Print out a space

Reset the string holding the dots and dashes back to an empty string.

Else if the current character is a space

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Reset the string holding the dots and dashes back to an empty string.

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Reset the string holding the dots and dashes back to an empty string.

Close the file holding the message to be translated

End

In the Main Function

Make a new Node that will be the beginning of the binary tree for decoding

Call decodeTreeConstructor with a string variable that is a file name that holds a Morse code table and the node variable that is the node created in main

Call E2M with the first string variable being a file name that holds a Morse code table and the second string variable being a file name that holds the message to be translated

Print a line

Call M2E with the node variable being the node created in main and the string variable being a file name that holds the message to be translated

Can call M2E and E2M as many times as necessary to carry out tests, should just follow structure above in order for clarity.

End

**Python(Pseudocode):**

This program translates messages from English to Morse code, as well as from Morse code to English using the coding language Python

Node class creates nodes that will contain a character value as well as a right and left child; both children are defaulted to none to be used to create the binary tree necessary for the Morse to English translation of a given message.

Function insert(self, path, letter)

Function inserts a letter into the binary tree at its correct position

If the first character in path is equal to (‘.’)

If the length of the path is greater than 1

Path will equal all elements of path besides the first element

If the node of the binary tree at the given position does not have a left child

Create left child with empty character value

Recursively call Insert function with the new path value

Else (meaning the path is 1 symbol)

If the node of the binary tree at the given position does not have a left child

Create left child with empty character value

Else

If the left node is empty

Add the given letter to the node

Else

Print an error message saying that a letter already exists in that node

Else if the first character in path is equal to (‘-‘)

If the length of the path is greater than 1

Path will equal all elements of path besides the first element

If the node of the binary tree at the given position does not have a right child

Create right child with empty character value

Recursively call Insert function with the new path value

Else (meaning the path is 1 symbol)

If the node of the binary tree at the given position does not have a right child

Create right child with empty character value

Else

If the right node is empty

Add the given letter to the node

Else

Print an error message saying that a letter already exists in that node

End

Function find\_letter(self, hidden\_letter\_path)

Function finds the character in the binary tree that corresponds to the Morse code given

If the first character in hidden\_letter\_path is equal to (‘.’)

If the length of the hidden\_letter\_path is greater than 1

hidden\_letter\_path will equal all elements of hidden\_letter\_path besides the first element

Returns a recursive call of find\_letter using the new hidden\_letter\_path and the left child of the previous node in the binary tree

Else

Returns the letter at the left child node from the current node in the binary tree

Elseif the first character in hidden\_letter\_path is equal to (‘-’)

If the length of the hidden\_letter\_path is greater than 1

hidden\_letter\_path will equal all elements of hidden\_letter\_path besides the first element

Returns a recursive call of find\_letter using the new hidden\_letter\_path and the right child of the previous node in the binary tree

Else

Returns the letter at the right child node from the current node in the binary tree

End

Function English2Morse

Function translates a file from English to Morse code, inputting the translation into a new file

Create an empty dictionary, 2 empty strings, and an empty array

Create a Boolean exception2 value defaulted to True

If the user inserted new Morse symbols

Open the new Morse table

Else

Open the default Morse table

For all lines in the Morse table file

Add the values in the line to the dictionary, with the first character being the key, and all characters after the spaces until the end of the line being the value for that given dictionary entry

Close the Morse table being used

While a valid file name needs to be entered

Try

Prompt the user for a file name to read in for translation

Open the file to be translated

Break from the loop

Except

Prompt the user by telling them the file does not exist and they can try again inputting a valid file

Creates a new file that will contain the message after it is translated from English to Morse

Invariant: line is always less than or equal to the number of lines in the English to Morse file

For all lines in the file being translated from

While the first character in a line equals a space (‘ ‘)

Write a space into the new file

Make the line equal all values of the line besides the first value

Split the line into individual words and insert into the array

If the array of words is not empty

If file read in already contains Morse values

Sets an exception2 value to false

Assigns an error message

Has shell print out an error message telling the user that the file contains no letters

Break from the loop

Else

# Invariant: word is always less than or equal to number of words in words list

For each word in the word list array

# Invariant: letter is always less than or equal to the number of letters in word

For each letter in each word

Try

If the letter is not uppercase

Based off the dictionary, using the letter as the key, add the values dots and dashes to a string, followed by a space

Else

Make the letter lowercase

Starting with a (‘\*’) symbol to detonate that the letter is uppercase, based off the dictionary, using the letter as the key, add the values dots and dashes to a string, followed by a space

Except (letter not found)

Boolean value created at the beginning is set to false

Add character to string of characters followed by (‘,’), the list being the characters that are not found in the Morse table

Reasoning for not translating letter is given (letter not found)

Write the translated word to the new file

Make the string that contained the translated word an empty string again.

Write 3 consecutive spaces into the new file to separate words (3 spaces instead of 1, because 1 space separated letters)

Make exceptions set to true

Write a new line to the translated file

Else

Write a new line to the translated file

Close the file that reads in the message to be translated

Close the file that was created to contain the translated message

If exception is true

If exception2 is true

Shell prints out a success message

Else

Shell prints out a partial success message, saying some symbols could not be found in the Morse table

Else

Shell prints out a failure message along with the reason why.

End

Function Morse2English

Function translates a file from Morse code to English, inputting the translation into a new file

Create an empty root node.

Create a count, defaulting to 1

Create a Boolean exception2 value defaulted to True

If the user inserted new Morse symbols

Open the new Morse table

Else

Open the default Morse table

For all lines in the Morse table file

Call the insert function. This will input the values from the Morse table into the correct position in a binary tree.

Close the Morse table being used

While a valid file name needs to be entered

Try

Prompt the user for a file name to read in for translation

Open the file to be translated

Break from the loop

Except

Prompt the user by telling them the file does not exist and they can try again inputting a valid file

Create a new file that will contain the message after it is translated from Morse to English

#Invariant: line is always less than or equal to the number of lines in hidden\_morse\_file

For all lines in the file being translated from

While the first character in a line equals a space (‘ ‘)

Write a space into the new file

Make the line equal all values of the line besides the first value

Split the line into individual letters (each set of letters is separated by one space)

Correct the last letter in the file

If array of Morse characters is not empty

If file contains Morse Values

# Invariant: hidden\_letter\_path is always less than or equal to the number of dashes and dots in morse\_letter

For each set of Morse Characters in the array of Morse Characters

Try

If the set of Morse characters is not empty

If the first character is not a (‘\*’)

Search the binary tree for the equivalent letter and write the letter into the new file

Else

Make a new string, excluding the first character

Search the binary tree for the equivalent letter and write it capitalized in the new file

Else

Increment the count up by 1

If the count is divisible by 2 with no remainders

Write to spaces to the file to indicate the end of the word

Else

Do nothing

Except

Shell prints out error saying Morse code was not found in the file

The Boolean expcetion2 is set to false

A reason is given for why the translation failed

Write a new line to the translated file

Set an exception value to be true

Else

Set an exception value to be false

Shell prints out error saying the file entered is not a Morse code file

A reason is given to why the translation failed

Break from the loop

Else

Write a new line to the translated file

Close the file that reads in the message to be translated

Close the file that was created to contain the translated message

If exception is true

If exception2 is true

Shell prints out success message

Else

Shell prints out a partial success message, saying some of the Morse code was ignored due to being invalid

Else

Shell prints out a failure message along with the reason why

End

Function newMorseTable

Function allows the user to change the symbols of Morse code table from dots and dashes to others

Declare 3 variables to be used

Set the first variable to be equal to (‘\*’)

While the first variable is equal to (‘\*’) or the length of the variable is greater than 1

First variable equals new variable inputted by user

If the first character of the variable is (‘\*’)

Shell tells user that the symbol must be different

Elseif the length of the variable is greater than 1

Shell tells user that the symbol can only be one character

Second variable is set to the value of the first variable

While the first and second variable are equal or the second variable is equal to (‘\*’) or the length of the second variable is greater than 1

Second variable equals new variable inputted by user

If the first variable equals the second variable

Shell tells user that the symbols must be different

ElseIf the first character of the variable is (‘\*’)

Shell tells user that the symbol must be different

Elseif the length of the variable is greater than 1

Shell tells user that the symbol can only be one character

Else

Do nothing

Opens the original Morse code table file

Creates new file for the new Morse code table

# Invariant: line is always less than or equal to the number of lines in the Morse Table file

For all lines in the Morse table file

Write the old Morse code into the new file (excluding the dots and dashes)

# Invariant: i is always a dash or a dot

For all elements in the line excluding the starting letter

If the element is a (‘.’)

Replace with the new first variable in the new file

Elseif the element is a (‘-‘)

Replace with the new second variable in the new file

Else

Do nothing

Write a new line in the new file

Close the original Morse code table file

Close the new Morse code table file

Set the newM to true, meaning you did change the symbols

Call the Enlgihs2Morse function

End

In the Main Function

Declare two variables. One takes the value (‘.’) while the other takes the value (‘-‘)

Declare a variable that holds the truth value of whether or not the user has changed the values for the Morse symbols. This variable should be defaulted to false.

Print a welcome message to the user (Hello, Welcome to the Morse Code Project)

##Add Invariant while \_\_\_\_\_\_

Prompt the user to choose what they would like to do with the program (Translate Morse to English, Translate English to Morse, change the given Morse code symbols, or exit the program)

If the user choses to translate a message from Morse to English

Call the function Morse2English

ElseIf the user choses to translate a message from English to Morse

Call the function English2Morse

Elseif the user choses to exit the program

Print a goodbye message

Break from the loop

Elseif the user choses to change the Morse code symbol(s)

Call the function newMorseTable

Else

Output an error message

End

-------------------------------------------------------------------------------------------------------------------------------

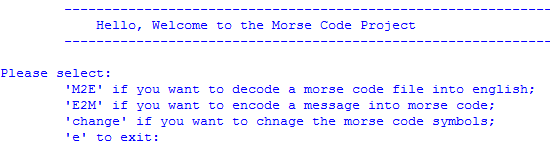
Section 4

Description of Example

-------------------------------------------------------------------------------------------------------------------------------

This will include images of the process of the program steps as they are running with blurbs describing what is happening.

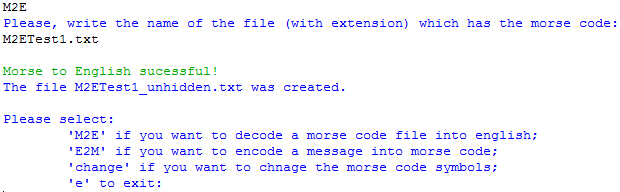
Python: The first step is selecting your operation. Typing “M2E” will open the morse to english functionality, while typing “E2M” will open the english to morse function. Typing “change” allows the user to change the morse code symbols, for use in special applications. Simply typing “e” will end and exit the program.



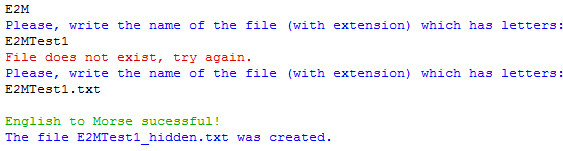
If change is typed, the following prompt is displayed:



If M2E is chosen, the user will be prompted to input the filename with extension. If the file is found to be valid, the English to morse decoding operation is completed on the file.



If E2M is chosen, the user will be prompted to input the filename with extension. If the file is found to be valid, the Morse to English encoding operation is completed on the file.



C++:

-------------------------------------------------------------------------------------------------------------------------------

References

-------------------------------------------------------------------------------------------------------------------------------

HISTORY. (2019). *Morse Code & the Telegraph*. [online] Available at: https://www.history.com/topics/inventions/telegraph [Accessed 23 Apr. 2019].

Worldkings - World Records Union. (2019). *WORLDKINGS - On This Day – January 06, 2019 - Samuel Morse Demonstrates The Telegraph, in 1838.*. [online] Available at: http://worldkings.org/news/world-almanac-event-academy/worldkings-on-this-day-january-06-2019-samuel-morse-demonstrates-the-telegraph-in-1838 [Accessed 23 Apr. 2019].