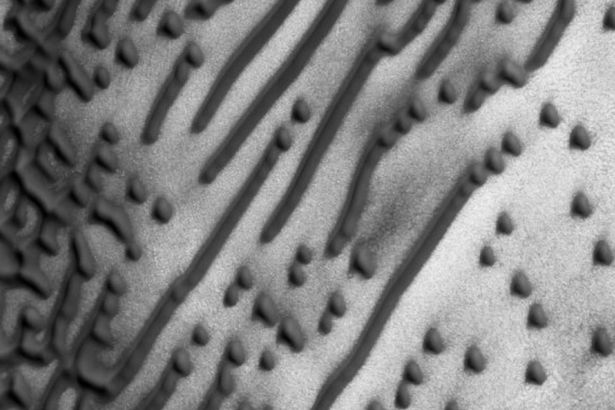
**Morse Code Encoder/Decoder**

*Data Structures, Project 3B*

*7/26/2016*

# team members: Gulnoza KhakimovA, Rebecca Peabody



*NASA/JPL/University of Arizona – Morse code like markings on the surface of Mars*

**Summary:** This project is a Morse Code encoder/decoder. Plain text and encoded text inputs are assigned to variables in Main( ) and are processed by member functions of the Morse\_Coder class to produce plain text or encoded text output.

Coding coordinated through GitHub, repository URL: <https://github.com/beccasbound/Project_3B.git>

Design/Implementation

The project is designed around the Morse\_Coder class, which has three main functions:

The **build\_morse\_tree( )** function processes an input file with lines made up of a letter of the alphabet followed by its Morse code representation. The letter is stored in a variable, which is assigned to the current node after a series of steps consisting of lefts (for dots) and rights (for dashes), with new nodes being added as needed.

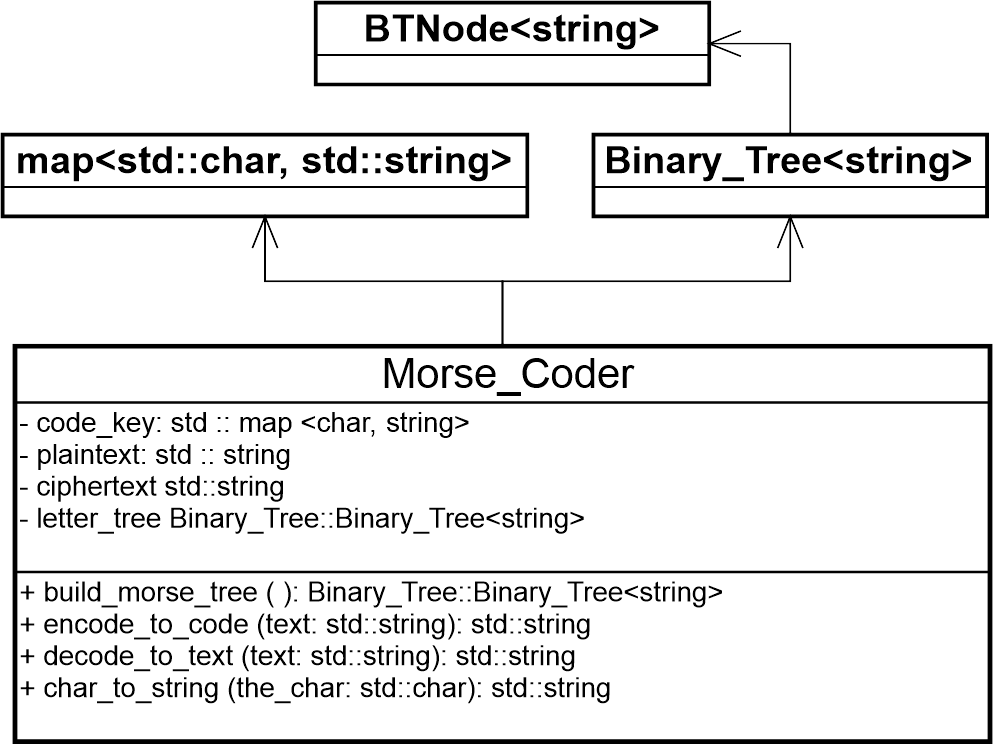
The **encode\_to\_code( )** function takes a plain text word as input and produces its Morse code equivalent. A map having letters as keys and Morse code for those letters as values is used in encoding.

The **decode\_to\_text( )** function takes a segment of Morse code in string form (ex: ..\_ .\_ \_...) and decodes it, returning a plain text word. The decoding function uses the binary tree built with the build\_morse\_tree( ) function, following the same strategy – left for dots and right for dashes, then reading the value of the resulting node, which is a letter in string form, and concatenating those letters together to form the ouput string.

Other classes used include Binary\_Tree and BTNode (see references).

The program is controlled through the function Main.cpp. Messages to be decoded or encoded must be entered in string form there. A user interface loop to allow a user to directly enter messages could be easily implemented.

**UML DIAGRAM:**

****

Performance

**BIG O ANALYSIS: Morse\_Coder class**

build\_morse\_tree() – O(n) (Each item in the input file must be processed to build the tree.)

char\_to\_str(char the\_char) – O(*1*)

encode\_to\_code(string text) – O(*n*)

decode\_to\_text(string text) – O(*n*)

Using a binary tree as the data structure to hold the key/code combination is an efficient way to accomplish decoding. Alternatively, a linear structure that contained a letter and its Morse code as values could be used, but would not be very efficient, since for each Morse “letter” in the input text, an iterative search would have to be performed on the data structure (say it has m items), and this would have to be repeated for each “letter” of the *n* item input, so the performance would be O(*nm*). Finding the right letter in the binary search tree is O(log *n*), but the overall performance of the decoder is O(*n*) because it is still necessary to process each letter of the input text.

Error Checking

Plain text input to be encoded should be a string consisting of letters of the Latin alphabet. Input that has non-letter characters results in an error message being printed to the console along with the program output.

Morse code input to be decoded should consist of a series of Morse code letter representations, separated by spaces. For each series of characters not separated by spaces, if the group is not a Morse code character, an error message is printed to the console along with the output.

References

C++ reference website: <http://www.cplusplus.com/> .

Class Notes, *UMKC CS303, Summer 2016*.

Binary\_Tree and BTNode classes, *author M. Kuhail*.