Reference Manual

Generated by Doxygen 1.8.8

Fri Nov 17 2017 07:11:56

Contents

1 Telegraph::decode

decodes morse code into english text

Parameters

`in,out`	*text*	A char array holding inputted or translated english text
`in,out`	*morse*	A char array holding inputted or translated morse code

Returns

void

2 Telegraph:encode

encodes an english message into morse code

Parameters

`in,out`	*text*	A char array holding inputted or translated english text
`in,out`	*morse*	A char array holding inputted or translated morse code

Returns

void

3 MorseCode

A struct holding letters of the alphabet along with numbers and special characters. Each symbol is associated with a morse code sequence,a max of 7 '.'s or '-'s total can be held in the sequence.

4 TNODE

A tree node to be used in creation of the binary tree. Used in conjunction with the open() function, when creating the tree, a tree node is made with '∗' as a default symbol and no child nodes. The symbol is overwritten if the sequence ends at that tree node and child nodes are created if the sequence forces a node to be made there.

5 Class: Telegraph

A class holding a table of size 40 consisting of symbols and their morse code equivalents. A root node is created automatically for creation of the binary tree. A DestroyTree() function exists for the closing process of the binary tree. The class has open() and close() functions for maintaining the morse code binary tree. The class also holds encode() and decode() functions for english to morse translations and vice versa.

6 Telegraph::open()

Creates and opens a binary tree holding english symbols and their morse code equivalents

A table holding english symbols and their morse code equivalent is read. For each symbol, the morse code sequence is read and a node pointer points through the nodes depending on the sequence, creating nodes if they don't exist. If a '.' is read, it moves into the left child node of the root. If a '-' is read, it moves into the right child node of the root. This keeps going until the end of the sequence. For example, '.–' would cause the node pointer to point to the left child node of the root, then the right child node of the node it was pointing to, and then the left child node again of the node it was pointing to. If the nodes didn't exist, they would be created. At the end of the sequence, the symbol associated with it is stored in the node and the node pointer goes back to root, restarting the process until the table is finished.

7 Telegraph::DestroyTree/Telegraph::close()

Traverses the binary tree, deleting each node until finally deleting the root node itself

The root node is passed in and begins the deletion process. The close function calls DestroyTree, a recursive function that will traverse the tree until it hits the end of both sides. Once that happens, the nodes are deleted until it reaches root. Root is then deleted, triggering the base case where a node does not exist, ending the function

Parameters

`in`	∗*node*	The root node of the binary tree

Returns

void

8 Specification

This is the Morse Code Translator. This lab utilizes a binary search tree to translate an English phrase into morse code (encode). It uses a similar tree to translate morse code into English (decode). The entire program is run on an html webpage that gives useful UI and a ease of navigation.

Features:

1) A user can encode or decode a message.

2) The message is clear and easy to read.

3) UI is very easy and clear to use.

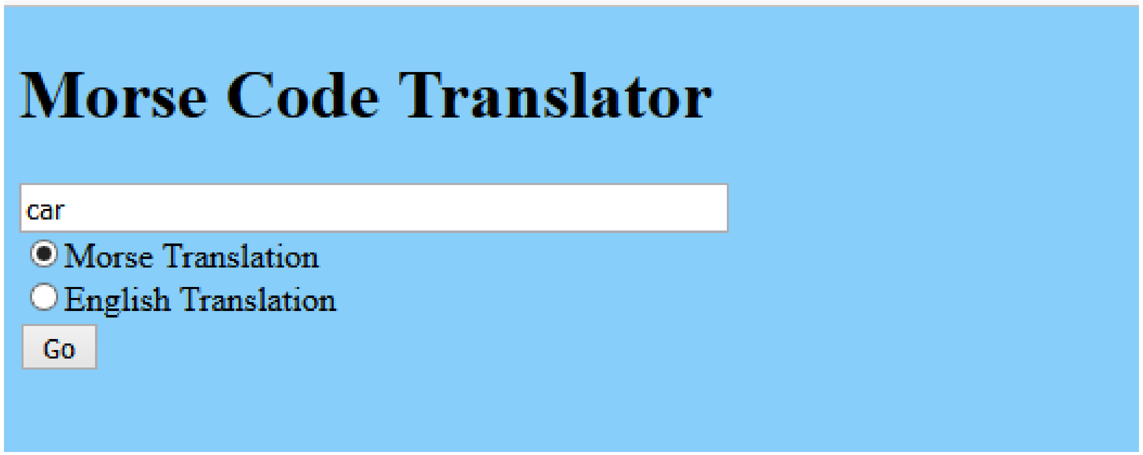
9 Analysis

When the user goes to the html page enters their information and clicks submit the program begins to run. Immediatly the translation based upon whatever the user selects will become available and the user can see it. The program is as simple as that. The code uses a binary search tree to find and create the translation. The Design section will explain this more thoroughly.

10 Design

Html was what we used to create the UI for this lab. THe translation program however was done in C++. We implemented a binary tree which basically allows us to store elements that can reference elements lower than it. Like tree -> branches -> twigs -> leaves. The idea is that only an element of lower value can be stored in bottom of another element and lower is on the left and greater on the right. So in our lab a . was considered lower and a - was conidered higher. This allowed us to reference letters within the tree very quickly.

11 Test: Morse 1



Morse Code Translator

car

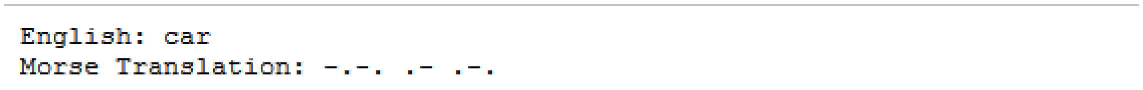
(●) Morse Translation

( ) English Translation

Go

Figure 1: Morse code to be decoded

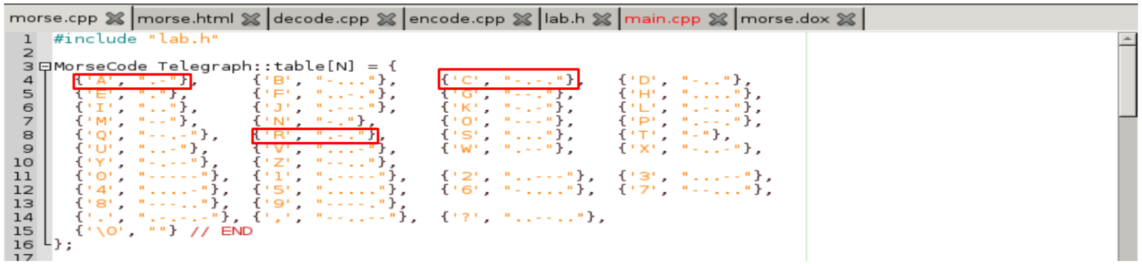
12 Test: Morse 2



```
English: car
Morse Translation: -.-. .- .-.
```

Figure 2: Morse code decoded

13 Test: Morse Verified

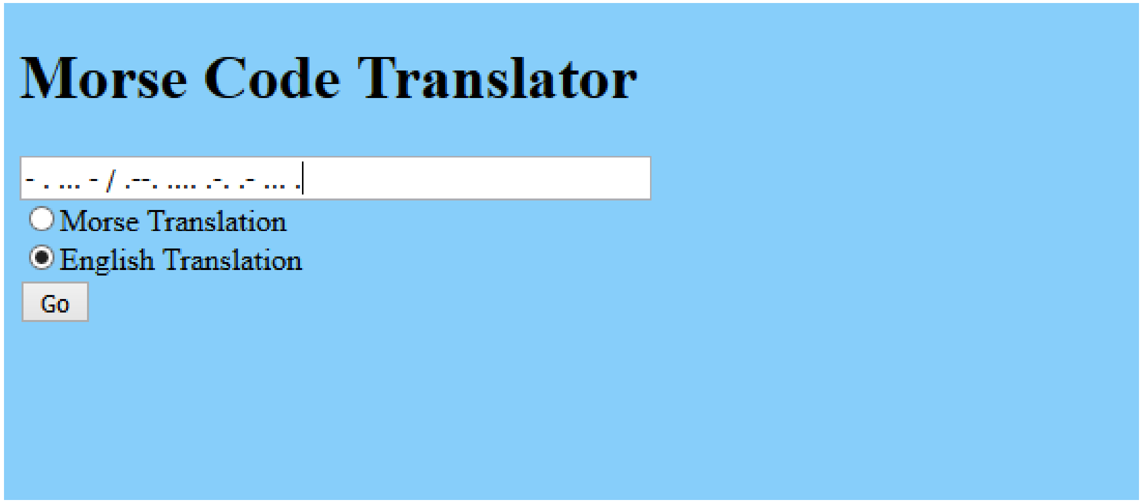


```
#include "lab.h"

MorseCode Telegraph::table[N] = {
    {'A', ".-"},          {'B', "-..."},        {'C', "-.-."},     {'D', "-.."},
    {'E', "."},           {'F', "..-."},        {'G', "--."},      {'H', "...."},
    {'I', ".."},          {'J', ".---"},        {'K', "-.-"},      {'L', ".-.."},
    {'M', "--"},          {'N', "-."},          {'O', "---"},      {'P', ".--."},
    {'Q', "--.-"},        {'R', ".-."},         {'S', "..."},      {'T', "-"},
    {'U', "..-"},         {'V', "...-"},        {'W', ".--"},      {'X', "-..-"},
    {'Y', "-.--"},        {'Z', "--.."},
    {'0', "-----"},       {'1', ".----"},       {'2', "..---"},    {'3', "...--"},
    {'4', "....-"},       {'5', "....."},       {'6', "-...."},    {'7', "--..."},
    {'8', "---.."},       {'9', "----."},
    {'.', ".-.-.-"},      {',', "--..--"},      {'?', "..--.."},
    {'\0', ""} // END
};
```

Figure 3: Morse code verified

14 Test: English 1

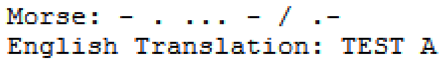


Morse Code Translator

```
- . ... - / .--. .... .-. .- ... .
```

Figure 4: English text to be encoded

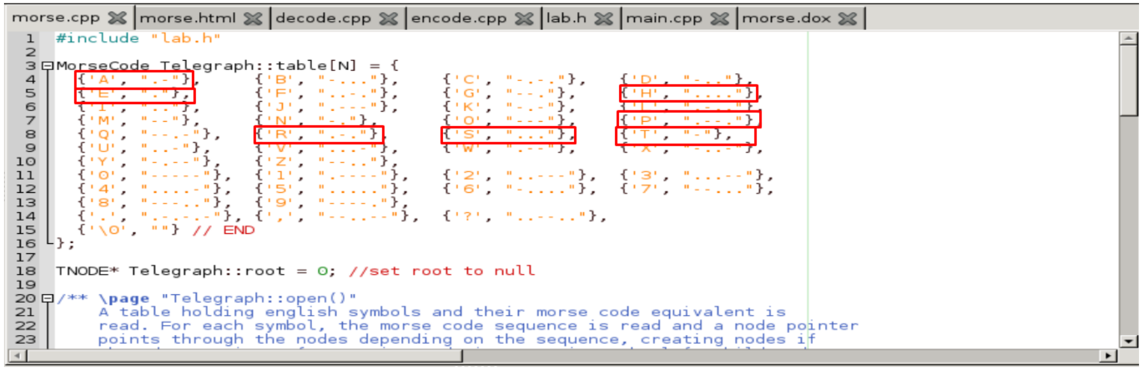
15 Test: English 2



```
Morse: - . ... - / .-
English Translation: TEST A
```

Figure 5: English text encoded

16 Test: English Verified


morse.cpp | morse.html | decode.cpp | encode.cpp | lab.h | main.cpp | morse.dox

```
#include "lab.h"

MorseCode Telegraph::table[N] = {
  {'A', ".-"},       {'B', "-..."},     {'C', "-.-."},     {'D', "-.."},
  {'E', "."},        {'F', "..-."},     {'G', "--."},      {'H', "...."},
  {'I', ".."},       {'J', ".---"},     {'K', "-.-"},      {'L', ".-.."},
  {'M', "--"},       {'N', "-."},       {'O', "---"},      {'P', ".--."},
  {'Q', "--.-"},     {'R', ".-."},      {'S', "..."},      {'T', "-"},
  {'U', "..-"},      {'V', "...-"},     {'W', ".--"},      {'X', "-..-"},
  {'Y', "-.--"},     {'Z', "--.."},
  {'0', "-----"},    {'1', ".----"},    {'2', "..---"},    {'3', "...--"},
  {'4', "....-"},    {'5', "....."},    {'6', "-...."},    {'7', "--..."},
  {'8', "---.."},    {'9', "----."},
  {'.', ".-.-.-"},   {',', "--..--"},   {'?', "..--.."},
  {'\0', ""} // END
};

TNODE* Telegraph::root = 0; //set root to null

/** \page "Telegraph::open()"
    A table holding english symbols and their morse code equivalent is
    read. For each symbol, the morse code sequence is read and a node pointer
    points through the nodes depending on the sequence, creating nodes if
```

Figure 6: English text verified

17 Morse Binary Tree

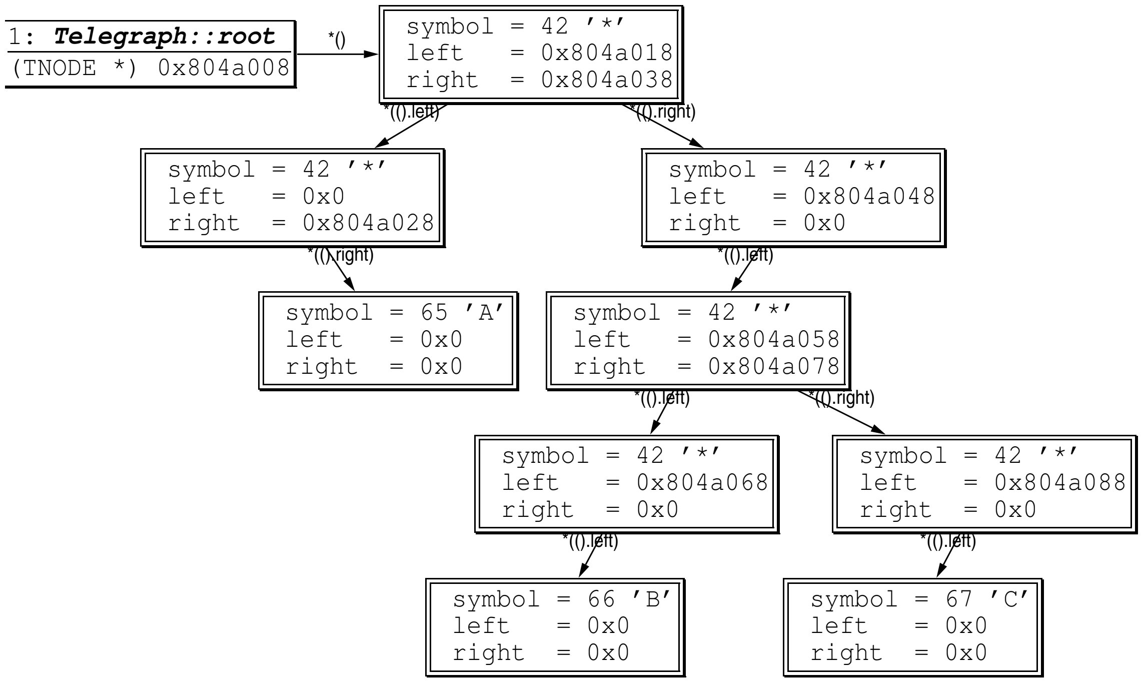


Figure 7: Morse Tree

18 HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- saved from url=(0043)http://localhost:8080/cs124/lab6/morse.html -->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" wtx-context="6D7991A1-0A29-456F-A1AE-A3E37871556A"
<title>CS 124 Lab 6 - MORSE CODE TRANSLATOR</title>

<meta name="generator" content="Geany 1.24.1">
</head>

<body bgcolor="#87CEFA">
   <h1>Morse Code Translator</h1>
   <form action="http://localhost:8080/cgi-bin/morse" wtx-context="3E5B8EA5-7244-4FCB-B3CB-CF9880D40BE8">
       <input type="text" name="e" size="50">
       <br>
       <input type="radio" name="o" value="morse">Morse Translation
       <br>
       <input type="radio" name="o" value="english">English Translation
       <br>
       <input type="submit" value="Go">
   </form>
   <!--img src="/images/graph.png"-->



</body></html>
```

19 Class Index

19.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

20 File Index

20.1 File List

Here is a list of all files with brief descriptions:

21 Class Documentation

21.1 MorseCode Struct Reference

```
#include <lab.h>
```

Public Types

- enum { N =7 }

Public Attributes

- char symbol
- char code [N]

21.1.1 Member Enumeration Documentation

21.1.1.1 anonymous enum

Enumerator

N

```
{N=7};
```

21.1.2 Member Data Documentation

21.1.2.1 char MorseCode::code[N]

21.1.2.2 char MorseCode::symbol

The documentation for this struct was generated from the following file:

- lab.h

21.2 Telegraph Class Reference

```
#include <lab.h>
```

Public Member Functions

- void encode (char text[ ], char morse[ ])
- void decode (char morse[ ], char text[ ])

Static Public Member Functions

- static void open ()
- static void close ()

21.2.1 Member Function Documentation

21.2.1.1 static void Telegraph::close ( ) [inline], [static]

```
{ DestroyTree(root); }
```

21.2.1.2 void Telegraph::decode (char *morse[ ],* char *text[ ]*)

```
{
    char *dd;
    int count = 0;
    TNODE *node;

    node = root;
    for(dd = morse; *dd; dd++) {
        if(*dd == '.')
            node = node->left;
        else if(*dd == '-')
            node = node->right;
        else if(*dd == '/') {
            node->symbol = ' ';
        }
        else {
            text[count] = node->symbol;
            node = root;
            count++;
        }
    }
}
```

21.2.1.3 void Telegraph::encode (char *text[ ],* char *morse[ ]*)

```
{
    int i;
    char c, *t, *dd; // t points to text
                     // dd points to a string of dots and dashes
    for(t = text; *t; t++)
    {

        c = toupper(*t);

        // If space, add a space to the morse string:
        if(c == ' ') {
            *morse++ = '/';
            *morse++ = ' ';
            continue;
        }

        // Find this symbol in the MORSECODE table
        // skip this symbol if not found:
        for(i = 0; table[i].symbol; i++)
            if(table[i].symbol == c) break;
        if(!table[i].symbol) continue;

        // Copy its code into the morse string:
        dd = table[i].code;

        while(*dd) *morse++=*dd++;

        // Add one space to separate letters:
        *morse++ = ' ';
    }
    *morse = '\0';
}
```

21.2.1.4 void Telegraph::open ( ) [static]

```
{
    char* dd;
    Telegraph::root = new TNODE;
    TNODE* node; TNODE* nextnode;
    for(int i = 0; i < N; i++) {
        node = root;
        for( dd = table[i].code; *dd; dd++) {

            if(*dd == '.')
            {
                nextnode = node->left;
                if(not nextnode)
                {
                nextnode = new TNODE;
                node-> left = nextnode;
                }
            }
            else if(*dd == '-')
             {
                nextnode = node->right;
                if(not nextnode)
                {
                nextnode = new TNODE;
                node-> right = nextnode;
                }
            }
            else std::cerr << "Unknown morse code" << std::endl;
            node = nextnode;
         } //not dash or dot, must be null so assign symbol
         node->symbol = table[i].symbol;
    }
}
```

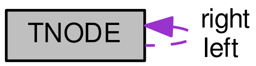
The documentation for this class was generated from the following files:

- lab.h
- decode.cpp
- encode.cpp
- morse.cpp

21.3 TNODE Struct Reference

```
#include <lab.h>
```

Collaboration diagram for TNODE:



Public Member Functions

- TNODE ()

Public Attributes

- char symbol

- TNODE ∗ left
- TNODE ∗ right

21.3.1 Constructor & Destructor Documentation

21.3.1.1 TNODE::TNODE ( ) [inline]

```
            {
        symbol = '*';
        left = 0;
        right = 0;
    }
```

21.3.2 Member Data Documentation

21.3.2.1 TNODE∗ TNODE::left

21.3.2.2 TNODE∗ TNODE::right

21.3.2.3 char TNODE::symbol

The documentation for this struct was generated from the following file:

- lab.h

22 File Documentation

22.1 decode.cpp File Reference

```
#include "lab.h"
```

22.2 encode.cpp File Reference

```
#include "lab.h"
```

22.3 lab.h File Reference

```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <sstream>
#include <algorithm>
```

Classes

- struct MorseCode
- struct TNODE
- class Telegraph

22.4 main.cpp File Reference

`#include "lab.h"`

Functions

- int main ()

22.4.1 Function Documentation

22.4.1.1 int main ( )

```
{
    Telegraph::open();
    Telegraph t;
    char text[80], morse[300];

    std::string s = getenv("QUERY_STRING");
    std::string message,ignore,option;
    std::stringstream ss;
    ss << s;

    while(ss) {
        getline(ss,ignore,'=');
        getline(ss,message,'&');
        getline(ss,option,'&');
    }

    std::replace(message.begin(),message.end(),'+',' ');

    for(int i = 0; i < message.length(); i++) {
        char c = message[i];
        if(c == '2')
            message.erase(i,2);
    }

    std::replace(message.begin(),message.end(),'%','/');

    if(option == "o=morse") {
        strcpy(text,message.c_str());
        t.encode(text,morse);
        std::cout << "English: " << message << std::endl;
        std::cout << "Morse Translation: " << morse << std::endl;
    }
    else if(option == "o=english") {
        message = message + "  ";
        strcpy(morse,message.c_str());
        t.decode(morse,text);
        std::cout << "Morse: " << message << std::endl;
        std::cout << "English Translation: " << text << std::endl;
    }
    Telegraph::close();
}
```

22.5 morse.cpp File Reference

```
#include "lab.h"
```

22.6 morse.dox File Reference

Index