LAB 6: MORSE CODE TREE BY:AMULDEEP DHILLON

Morse Code Lab

The purpose of this lab is to explore techniques to build and search binary trees. This is an efficient way to get the best properties of arrays and linked lists combined. In this case we will be using binary search tree to make a Morse Code translator.

This will allow us to search through data at greater efficieny with simplicity

Morse header File

Contains class for:

- Morse
- Tnode

and shared variables

```
#pragma once

const int N = 10; //max morse code size
struct Morse
{
    char symbol;
    char code[N];
};

struct Tnode{
    char symbol;
    Tnode* left;
    Tnode* right;
    Tnode(){symbol = '*';
        left = right = 0;
    }
};
```

Telegraph header file

Contains the class: Telegraph

```
#pragma once
#include "Morse.h"
#include <iostream>

const char DOT = '.';
const char DASH = '-';
class Telegraph
{
private:
    static Morse table[40];
    static Tnode * root;
    static void destroyTree(Tnode *node);
    static void addTnode(char c, Tnode* node, Tnode *& nextnode);
public:
    static void buildTree();
    static void destroyTree();
    void encode(char text[], char morse[]);
    void decode(char morse[], char text[]);
};
```

Morse Code Array

```
#include "Telegraph.h"
#include "Morse.h"
Morse Telegraph::table[] = {
  {'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"}
};
```

Contains an array of type Morse that holds every character of morse code with it corresponding translation

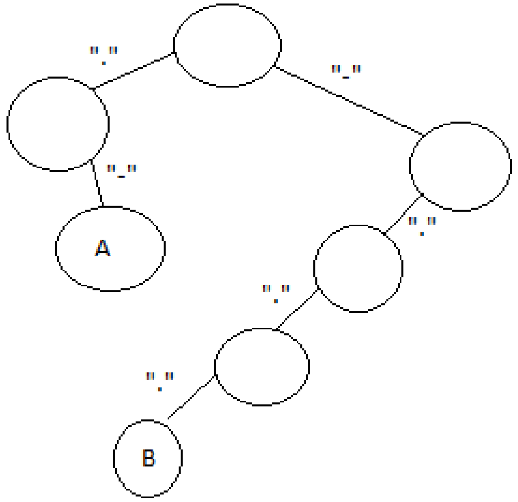
Build Tree

```
//#include <cctype>
#include "Telegraph.h"
#include "Morse.h"

Tnode * Telegraph::root = 0;
void Telegraph::buildTree()
{
    Tnode *node, *nextnode;
    int i; char *dd;
    root = new Tnode;    if (!root) return;
    root->symbol = ' ';
    for (i = 0; table[i].symbol; i++)
    {node = root;
        for(dd = table[i].code; *dd; dd++)
        {
            addTnode(*dd,node,nextnode);
            node = nextnode;
        }
        node->symbol = table[i].symbol;
    }
}
```

Creates a Tree by:

- Making a pointer named root to a Tree node
- The root is left blank
- Then it goes through the Morse Array named table placing pieces
- So for ”A” it goes to the left followed by a right and places A
- Then for ”B” it goes right,left,left,left and places B
- This repeats for all characters in table



Add T Node

```
#include "Telegraph.h"
void Telegraph::addTnode(char c, Tnode* node, Tnode*& next)
{
    if(c == DOT)
    {
        next = node->left;
        if(not next)
        {
            next = new Tnode;
            node->left=next;
        }

    }
    else if(c == DASH)
    {
        next = node->right;
        if(not next)
        {
            next = new Tnode;
            node->right=next;
        }
    }
}
```

Adds a Tree Node by:

- Checking if the element in array table is a dot or dash
- If it is a dot then it makes a branch to the left and adds a Tree node to the end
- If it is a dash then it makes a branch to the right and adds a Tree node to the end

Encode

Converts text into morse code by:

- Turning each character to uppercase
- Adding spaces where spaces are located
- Looks at the Morse array and finds the matching character
- Adds the corresponding symbols to the message and adds a space

```
#include "Telegraph.h"
void Telegraph::encode(char text[], char morse[])
{
    int i;
    char c, *t, *dd;
    for(t = text; *t; t++){
        c = toupper(*t);
        if (c==' '){
            *morse++ = ' '; continue;}
        for(i=0;table[i].symbol; i++)
            if(table[i].symbol == c)break;
            if(!table[i].symbol)continue;

            dd = table[i].code;
            while (*dd) *morse++ = *dd++;
            *morse++ = ' ';
        }
        *morse = '\0';
    }
```

Decode

```
#include "Telegraph.h"
void Telegraph::decode(char morse[], char text[]){
    char *dd;  Tnode * node;  node = root;
    for(dd = morse; *dd; dd++){
        if(*dd == DOT)
        {
            node = node -> left;
        }
        else if(*dd == DASH)
        {
            node = node -> right;
        }
        else if(*dd == ' ')
        {
            *text++ = node->symbol;
            *text++ = ' ';
            node = root;
            continue;
        }
    }
    *text = '\0';
}
```

Converts morse code to English by:

- Looking at each character of the message
- If it is a dot, then it moves down the left branch
- It it is a dash, then it moves down the right branch
- If it is a space, then it finds the current Tree node's symbol and adds that to the text while adding a space and returning to root
- Because of this when it encounters a space between words it adds two spaces since the current node at space will be root whose symbol is a space

Destroy Tree

```
#include "Telegraph.h"
void Telegraph::destroyTree(){
    destroyTree(root);
    root = 0;
}

void Telegraph::destroyTree(Tnode * node){
    if(node){
        destroyTree(node->left);
        destroyTree(node->right);
        delete node;
    }
}
```

Destroys a Tree with no parameters by:

- calling DestroyTree with one parameter
- and setting root to zero

Desroyes a Tree with one parameter by:

- Checks if there is a node
- If there is then it destroys the left and right path recursivly
- Then it deletes the final node itself

Lab cpp File

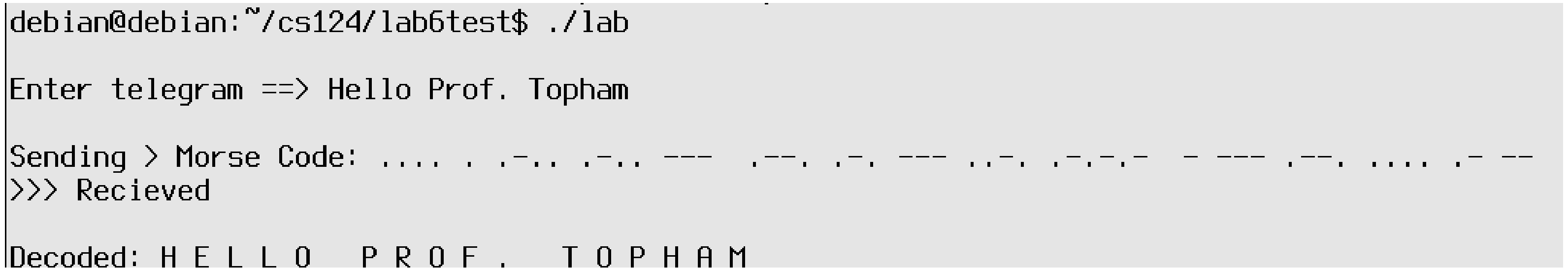
```
#include <iostream>
#include "Telegraph.h"
int main()
{
    Telegraph station;
    char text[80], morse[600];
    Telegraph::buildTree();
    std::cout << "\nEnter telegram ==> ";
    std::cin.getline(text, 80);
    std::cout << "\nSending > ";
    station.encode(text,morse);
    std::cout << "Morse Code: " << morse;
    std::cout << " >>> Recieved \n\n";
    station.decode(morse,text);
    std::cout << "Decoded: " << text << std::endl;

    Telegraph::destroyTree();

}
```

Executes all finctions and asks for message input then displays message in morse code, and re translated message

Test



```
debian@debian:~/cs124/lab6test$ ./lab

Enter telegram ==> Hello Prof. Topham

Sending > Morse Code: .... . .-.. .-.. ---  .--. .-. --- ..-. .-.-.-  - --- .--. .... .- --
>>> Recieved

Decoded: H E L L O   P R O F .   T O P H A M
```

Since the morse code table only has capital letters, all letters in the translation are capital