CS 232 - P4

Programmer Manual

By: Jeffrey Marron for B. Streller’s Spring 2016 CS232 class

Contents:

I. cs232\_p4\_client\_driver.cpp

II. huffman.h

I. - cs232\_p2\_client\_driver.cpp

The driver first initializes the main encoding tree / file system and then enters a switch case command system operated via user input. The functions mostly utilize the characterType structArray and the inFile name to pass values from int main to the main file system.

II. - huffman.h

This file specifies the definitions and implementations of the huffman class, the HuffTreeNode, characterType, and compare structs.

void print(struct HuffTreeNode\* root, string str)

A recursive function which traverses the huffman tree. If it follows root→left it adds “0” and if it follows root →right it adds “1” to string str. This function is only used for printing the encoded values directly from the tree itself.

string Graph::getFileName(ifstream &, string & inFilename)

Gets the input file name from the user.

void Huffman::encode(struct HuffTreeNode\* root, string str)

Creates a temporary characterType node of tempNode. Traverses the tree in the same manner as the print function however it assigns the values of the huffTreeNodes to the tempNode and then pushes the tempNode into the encodeTable vector.

void Huffman::countChars(ifstream& inFile, characterType array[], int size)

Parses the input file into an array which keeps track of the occurrences of each character from the input file.

void Huffman::createEncodeFile()

Traverses the encodeTable vector and prints the character value and the encoded value to an output file specified by the user.

void Huffman::printConsole()

Traverses the encodeTable vector and prints the character value and the encoded value to the console.

void Huffman::prioritize(characterType structArray[], int size)

Takes the characterType array and parses it into a HuffTreeNode if the structArray value is alphabetical and if the array has a non zero value. It then pushes this value into the priority queue PQ. It then takes two values popped from the queue and assigns them to the left and right and then creates a new HuffTreeNode with a ‘$’ as a delimiter character and then pushes that node into the queue. This runs until the priority queue’s size is not 1 which will then leave the root node. The encode function is then called on the top node to encode the tree.

void Huffman::createCodedFile(ifstream& inFile)

Asks the user to input a coded file name. Resets the infile used to create the table to the beginning. Then the encodeTable vector is traversed until its character value is equal to the array value of string temp which is pulled from the inFile. It then outputs the code value from the vector to the output file.

void Huffman::decode(ifstream& inFile)

Gets a coded input file and then parses it into string temp. String letter is then pushed back the value of the character at temp’s given index. The encodeTable vector is then traversed. If the code value is equal to string letter, then the character stored in the encodeTable character is printed to the console. Letter is then cleared.

void Huffman::loadCodeTable(ifstream& inFile)

Parses an input code table file into a characterType tempNode. This node is then pushed back into the encodeTable vector.