PROG2100: Morse Code

Glogowski,Joseph

W0453648

PROG2100

Assignment 4



Table of Contents

[IPO Chart 0](#_Toc158827389)

[Source Code 1](#_Toc158827390)

[Runtime Screen Captures 5](#_Toc158827391)

[References 6](#_Toc158827392)

# IPO Chart

|  |  |  |
| --- | --- | --- |
| Input | Processes | Output |
| * 1 or 2 depending on what translation you want * 1 or 2 depending on if itsd from a file or from the terminal * If its from the terminal, you would then input the string | userMenu()  displays welcome message  do{  asks user if they want morse to char or char to morse.  Then asks the user if its from the terminal or from a file.  Depending on the type of translation and input type, sets a variable to the corresponding number for a switch.  }  userInput()  asks user for input, could be English, could be morse.  readFile()  opens the file.  Reads each line and appends to the fileContent variable.  Once its at the end it ends the function.  fromText()  for each char in string.  Find the key in the map and translate to the value.  If it cant find it, appends a space.  Return morseCode.  fromMorse()  splits morse code into morse code symbol.  For each morse code symbol in morseCode.  If morse symbol is space, appends space to English.  If char is in the map, translate to morse. Else, append ‘?’.  Return english  displayOutput()  displays the output |  |

# Source Code

Main()

//files i will include  
#include "functions.h"  
  
//Title: Morse Code Translator  
//Author: Joe Glogowski  
//Date: November 23, 2023  
//Filename: morseCode.cpp  
//Purpose: Assignment 4  
  
#ifndef MORSECODE\_MAIN\_CPP  
#define MORSECODE\_MAIN\_CPP  
  
  
int main() {  
  
 //both maps for morse to char, and char to morse  
 std::map<char, std::string> charToMorse = **{**{'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', "--.."},{' ', " "},{' ', " "}**}**;  
 std::map<std::string, char> morseToChar = **{**{".--", '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'},{" ", ' '},{" ", ' '}**}**;  
  
 int translationType;  
 std::string userString;  
 std::string userIn;  
 std::string outputString;  
 std::string fileString;  
  
 //function to prompt user for type of translate  
 translationType = userMenu();  
  
 //function to read file and set to a variable  
 switch (translationType) {  
 //case 1 is from text, from the terninal  
 case 1:  
 userString = userInput();  
 outputString = fromText(userString, charToMorse);  
 displayOutput(outputString);  
  
  
 break;  
 //case 2 is from text, from a file  
 case 2:  
 //function to connect, read, and assign the file contents to a variable  
 fileString = readFile();  
 //use the fromText function to convert to morse code  
 outputString = fromText(fileString, charToMorse);  
 displayOutput(outputString);  
 //displayOutput  
  
 break;  
 //case 3 is from morse code, from the terminal  
 case 3:  
 userString = userInput();  
 outputString = fromMorse(userString, morseToChar);  
 displayOutput(outputString);  
  
  
 break;  
 case 4:  
 fileString = readFile();  
 outputString = fromMorse(fileString, morseToChar);  
 displayOutput(outputString);  
  
 break;  
 }  
  
  
  
  
  
  
 return 0;  
}  
  
#endif // MORSECODE\_MAIN\_CPP

functions.h

//  
// Created by W0453648 on 2023-11-23.  
//  
  
  
#ifndef MORSECODE\_FUNCTIONS\_H  
#define MORSECODE\_FUNCTIONS\_H  
  
#include <string>  
#include <map>  
  
  
int userMenu();  
  
std::string userInput();  
  
std::string readFile();  
  
std::string fromText(const std::string& userString, const std::map<char, std::string>& charToMorse);  
  
std::string fromMorse(const std::string& userString, const std::map<std::string, char>& morseToChar);  
  
void displayOutput(const std::string& displayOutput);  
  
  
#endif //MORSECODE\_FUNCTIONS\_H

Functions.cpp

//  
// Created by W0453648 on 2023-11-23.  
//  
  
#include "functions.h"  
  
  
#include <iostream>  
#include <map>  
#include <windows.h>  
#include <limits>  
#include <fstream>  
#include <sstream>  
  
int userMenu() {  
 int x;  
 int y;  
 int typeOfTranslate = 0;  
  
 std::cout << "Welcome to morse code translator!!!!" << std::endl;  
 Sleep(1000);  
  
 do {  
 std::cout << "Would you like to translate from text to morse code, or morse code to text." << std::endl  
 << "\t1 = from text to morse code." << std::endl << "\t2 = from morse code to text" << std::endl;  
 std::cin >> x;  
 if (x == 1){  
 std::cout << "Would you like to translate from the terminal or from a file" << std::endl  
 << "\t1 = from terminal" << std::endl << "\t2 = from file" << std::endl;  
 std::cin >> y;  
 if (y == 1){  
 //type of translation 1 is from text, from the terminal.  
 typeOfTranslate = 1;  
 std::cout << "Give me your text: " << std::endl;  
 }else if (y == 2) {  
 //type of translation 2 is from text code, from a file.  
 typeOfTranslate = 2;  
 }else{  
 std::cout << "Incorrect input, Please Try again" << std::endl;  
 continue;  
 }  
 }else if(x == 2) {  
 std::cout << "Would you like to translate from the terminal or from a file" << std::endl  
 << "\t1 = from terminal" << std::endl << "\t2 = from file" << std::endl;  
 std::cin >> y;  
 if (y == 1){  
 //type of translation 3 is from morse code, from the terminal  
 typeOfTranslate = 3;  
 std::cout << "Give me your morse code (one space between each morse-coded letter, and three spaces between each morse-coded word)" << std::endl;  
 }else if (y == 2) {  
 //type of translation 4 is from morse code, from a file  
 typeOfTranslate = 4;  
 }else{  
 std::cout << "Incorrect input, Please Try again" << std::endl;  
 continue;  
 }  
 }else{  
 std::cout << "Incorrect input, Please Try again" << std::endl;  
 continue;  
 }  
  
 }while(typeOfTranslate == 0);  
 return typeOfTranslate;  
}  
  
std::string userInput(){  
  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
  
 std::string userString;  
 std::getline(std::cin, userString);  
  
 return userString;  
}  
  
std::string readFile () {  
 std::ifstream file("fileToRead.txt");  
  
  
 if(!file.is\_open()){  
 std::cerr << "error opening file: " << "fileToRead.txt" << std::endl;  
 return "";  
 }  
  
 std::string fileContent;  
 std::string line;  
  
 while (std::getline(file, line)) {  
 fileContent += line + "\n";  
 }  
  
 file.close();  
  
 return fileContent;  
}  
  
std::string fromText(const std::string& userString, const std::map<char, std::string>& charToMorse) {  
 std::string morseCode;  
  
 for (char c : userString) {  
 auto it = charToMorse.find(c);  
 if (it != charToMorse.end()) {  
 morseCode += it->second + ' ';  
 }else {  
 morseCode += " ";  
 }  
 }  
 return morseCode;  
}  
  
std::string fromMorse(const std::string& morseCode, const std::map<std::string, char>& morseToChar) {  
 std::string english;  
 std::istringstream iss(morseCode);  
  
 std::string morseSymbol;  
 while (iss >> morseSymbol) {  
 if (morseSymbol == " ") {  
 english += ' '; // Handle space between words  
 } else {  
 auto it = morseToChar.find(morseSymbol);  
 if (it != morseToChar.end()) {  
 english += it->second;  
 } else {  
 english += '?';  
 }  
 }  
 }  
 return english;  
}  
  
void displayOutput(const std::string& outputString){  
  
 std::cout << "the output is: " << outputString << std::endl;  
  
}

# Runtime Screen Captures

A computer screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

# References

*C++ Tutorial*. (2019). W3schools.com. <https://www.w3schools.com/cpp/>

*cppreference.com*. (2019). Cppreference.com. <https://en.cppreference.com/w/>