/\*\*

\* This program can encode a message in a picture.

\* @author Tom Lowry

\* @verison

\* File: PA4.cpp

\* Created April 7, 2018

\* (c) Copyright Cedarville University, its Computer Science faculty, and the

\* authors. All rights reserved.

\* Summary of Modifications:

\* April 7, 2018 - Created the program

\*

\* Description: This program allows one to use steganography.

\*/

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include "LodePNG.h"

using namespace std;

//This makes sure that this is a valid file

bool checkPath(string path) {

fstream myfile;

myfile.open(path.c\_str());

if (myfile.is\_open()) {

myfile.close();

return true;

}

else {

return false;

}

}

//This is used to output an error if the arguements don't match up

int inputError(string arguement) {

cerr << "Does not recognize " << arguement << "\n"

<< "steg -e <original image name> <modified image name> [input ASCII text file name] \n"

<< "or \n"

<< "steg -d <modified image name> [output ASCII text file name] \n" << endl;

return -1;

}

//This reads in the file line by line and puts it into a vector of unsigned chars

vector<unsigned char> readTextFile(char\* path) {

string message;

string tempMessage;

ifstream file(path);

while (getline(file, tempMessage)) {

message += tempMessage + "\n";

}

vector<unsigned char> charMessage(message.begin(), message.end());

return charMessage;

}

//This encodes the message into the picture by changing the last bit

void encodeMessage(vector<unsigned char> &image, vector<unsigned char> message) {

//Checks to make sure message can fit in the picture

if (image.size() < ((message.size()) \* 8)) {

cerr << "This message is to large for this picture" << endl;

return;

}

//This sents all the lasts bits to zero so that they can be changed

unsigned char bit[8];

for (int i = 0; i < image.size(); i++) {

image[i] &= 0xFE;

}

for (int i = 0; i < message.size(); i++) {

//This makes 8 bits to put at the end

for (int j = 0; j < 8; j++) {

bit[j] = ((message[i] >> j) & 0x01);

}

//This makes sure the bits get put in the right order

for (int j = 0; j < 8; j++) {

//Need to put in the each of the 8 spots plus the offset of j plus

image[(i \* 8) + (j)] |= (bit[7 - j]);

}

}

return;

}

//This decodes the message in the picture

string decodeMessage(vector <unsigned char> image) {

string message;

unsigned char byte = 0x00;

for (int i = 0; i < image.size(); i++) {

byte |= (image[i] & 0x01);

if (i % 8 == 7) {

message += byte;

//Makes sure that it stops at the end of the word

if (byte == '\0') {

return message;

}

byte = 0x00;

}

byte = byte << 1;

}

return message;

}

//This prints out an error if the numbers argc is wrong

int arguementAmountError() {

cerr << "Please eneter the correct amount of inputs! Examples below\n"

<< "steg -e <original image name> <modified image name> "

<< "[input ASCII text file name] \n"

<< "or \n"

<< "steg -d <modified image name> [output ASCII text file name] \n"

<< endl;

return -1;

}

int main(int argc, char\* argv[]) {

//Error checking for the corrrect amount of arguements

if (argc > 5 || argc < 3) {

return arguementAmountError();

}

//Checks that the rest of the inputs are valid files

for (int i = 2; i < (argc); ++i) {

if (!checkPath(argv[i])) {

return inputError(argv[i]);

}

}

//This deals with all the encoding

if (string(argv[1]) == "-e") {

//This chooses whether to read in from a file or stdin

vector<unsigned char> text;

string tempText;

if (argc == 5) {

text = readTextFile(argv[4]);

}

else if(argc == 4) {

getline(cin, tempText);

copy(tempText.begin(), tempText.end(), back\_inserter(text));

}

else {

return arguementAmountError();

}

cout << "Encrypting your image... " << endl;

//This reads in the image

vector<unsigned char> image;

unsigned width, height;

unsigned error = lodepng::decode(image, width, height, argv[2]);

//if there's an error, display it

if (error) std::cout << "decoder error " << error << ": " <<

lodepng\_error\_text(error) << std::endl;

//This decodes the message in the image

encodeMessage(image, text);

//Encode the image

error = lodepng::encode(argv[3], image, width, height);

//if there's an error, display it

if (error) std::cout << "encoder error " << error << ": " <<

lodepng\_error\_text(error) << std::endl;

}

//This deals with the decoding

else if (string(argv[1]) == "-d") {

cout << "Decrypting your image... " << endl;

//This reads in the image

vector<unsigned char> image;

unsigned width, height;

unsigned error = lodepng::decode(image, width, height, argv[2]);

//if there's an error, display it

if (error) std::cout << "decoder error " << error << ": " << lodepng\_error\_text(error) << std::endl;

string hiddenMessage = decodeMessage(image);

//If a there is a third argument then it stores it in the file

if (argc == 4){

ofstream myFile;

myFile.open(argv[3]);

myFile << hiddenMessage;

myFile.close();

}

else{

cout << hiddenMessage << endl;

}

}

//This deals if the first arguement is wrong.

else {

return inputError(argv[1]);

}

return 0;

}