/\*\*

\* FileIO.h

\* Contains classes to facilitate reading and writing to files.

\*

\* @author Jeff Couch (couch009@cougars.csusm.edu)

\* @version 1.1 (09.18.2014)

\*\*/

#include <vector>

#include <string>

#include <fstream>

using namespace std;

/\*\*

\* Capital class

\*

\* A class for recording the position of capital letters in the messages

\* to be encrypted and decrypted.

\*\*/

class Capital

{

/\*\*

\* Class member variables

\* int x -- holds the vector index of each string containing a capital letter.

\* int y -- holds the string index of each capital letter within the vector.

\*\*/

private:

int x;

int y;

public:

/\*\*

\* Constructor for Capital class

\* Sets x and y values for created Capital object

\*

\* @param int xIn

\* @param int yIn

\*\*/

Capital(int xIn, int yIn);

/\*\*

\* Accessor for int x

\*

\* @return x

\*\*/

int getX();

/\*\*

\* Accessor for int y

\*

\* @return y

\*\*/

int getY();

};

/\*\*

\* FileIO class

\*

\* A class with static methods for reading and

\* writing files.

\*\*/

class FileIO

{

public:

/\*\*

\* Vector of Capital objects

\* Records the location of capital characters in the

\* string being read.

\*\*/

static vector<Capital> caps;

/\*\*

\* Reads the contents of a file (specified by fileName),

\* converts it to lowercase characters,

\* stores the location of any capital characters in a vector of

\* Capital objects, and places into a vector &returned.

\*

\* @param string fileName

\* @param vector<string> &returned

\*\*/

static void readFile(string fileName, vector<string> &returned);

/\*\*

\* Reads the contents of a vector, replaces any uppercase characters,

\* and writes the elements to a text file specified by fileName.

\*

\* @param string fileName

\* @param vector<string> &returned

\*\*/

static void writeFile(string fileName, vector<string> &returned);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* FileIO.cpp

\*

\* @author Jeff Couch (couch009@cougars.csusm.edu)

\* @version 1.1 (09.18.2014)

\*\*/

#include "FileIO.h"

//Vector for recording the location of capital letters.

vector<Capital> FileIO::caps;

void FileIO::readFile(string fileName, vector<string> &returned)

{

fstream fin;

fin.open(fileName.c\_str());

string tempString = "";

//Initialize the returned and Capital vectors.

returned.clear();

caps.clear();

//Reads in each string from the file and pushes it

//into the return vector.

while(fin)

{

fin >> tempString;

returned.push\_back(tempString);

if(fin.eof())

break;

}

//Checks for uppercase characters, records their position,

//and replaces them with lowercase characters.

for(int x = 0; x < returned.size(); x++)

{

for(int y = 0; y < returned[x].size(); y++)

{

if(returned[x][y] > 64 && returned[x][y] < 91)

{

caps.push\_back(Capital(x, y));

returned[x][y] = tolower(returned[x][y]);

}

}

}

}

void FileIO::writeFile(string fileName, vector<string> &toWrite)

{

fstream fout;

//Opens stream to file. If file does not exist, it is created.

fout.open(fileName.c\_str(), fstream::out);

//Replaces all uppercase characters in vector.

for(int x = 0; x < caps.size(); x++)

{

toWrite[caps[x].getX()][caps[x].getY()] =

toupper(toWrite[caps[x].getX()][caps[x].getY()]);

}

//Writes contents of vector to file.

for(int x = 0; x < toWrite.size(); x++)

{

if(x == toWrite.size()-1)

fout << toWrite[x];

else

fout << toWrite[x] << " ";

}

}

Capital::Capital(int xIn, int yIn)

{

x = xIn;

y = yIn;

}

int Capital::getX(){return x;}

int Capital::getY(){return y;}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* Crypt.h

\* class Crypt

\* Provides methods for encrypting and decrypting the contents of a vector of strings.

\*

\* @author Jeff Couch (couch009@cougars.csusm.edu

\* @version 1.1 (09.18.2014)

\*\*/

#include <vector>

#include <string>

#include <fstream>

using namespace std;

class Crypt

{

public:

/\*\*

\* Static method for encrypting a vector of strings by rotation.

\*

\* @param const vector<string> &unencrypted -- the vector to be encrypted

\* @param vector<string> &encrypted -- the contents of the vector are placed here

\* once they are encrypted.

\* @param int rotation -- the amount each character should be rotated for encryption

\*\*/

static void encrypt(const vector<string> &unencrypted, vector<string> &encrypted, int rotation);

/\*\*

\* Static method for encrypting a vector of strings by cryptogram.

\*

\* @param const vector<string> &unencrypted -- the vector to be encrypted

\* @param vector<string> &encrypted -- the contents of the vector are placed here

\*\*/

static void encrypt(const vector<string> &unencrypted, vector<string> &encrypted);

/\*\*

\* Static method for decrypting a vector of strings by rotation.

\*

\* @param const vector<string> &unencrypted -- the contents of the encrypted vector are placed

\* here once they have been decrypted.

\* @param vector<string> &encrypted -- the vector to be decrypted.

\* @param int rotation -- the amount each character should be rotated for decryption

\*\*/

static void decrypt(const vector<string> &encrypted, vector<string> &unencrypted, int rotation);

/\*\*

\* Static method for decrypting a vector of strings by cryptogram.

\*

\* @param const vector<string> &unencrypted -- the contents of the encrypted vector are placed

\* here once they have been decrypted.

\* @param vector<string> &encrypted -- the vector to be decrypted.

\*\*/

static void decrypt(const vector<string> &encrypted, vector<string> &unencrypted);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* Crypt.cpp

\*

\* @author Jeff Couch (couch009@cougars.csusm.edu

\* @version 1.1 (09.18.2014)

\*\*/

#include "Crypt.h"

/\*\*

\* Reads cryptogram from text file and returns it as a string.

\*

\* @param string fileName

\* @return string

\*/

string getKey(string fileName);

//Unencrypted representation of the alphabet to be used for encrypting and

//decrypting with cryptogram methods.

string alphabet = "abcdefghijklmnopqrstuvwxyz";

void Crypt::encrypt(const vector<string> &unencrypted, vector<string> &encrypted, int rotation)

{

encrypted.clear();

string tempString;

//Ensure that rotation amount is less than the length of the entire alphabet.

if(rotation > 26)

rotation = rotation % 26;

//Iterate over each string in the vector.

for(int x = 0; x < unencrypted.size(); x++)

{

tempString.clear();

//Iterate over each character in the current string.

for(int y = 0; y < unencrypted[x].size(); y++)

{

//Check if current character is a valid lowercase character, encrypt it, and

//append it to the temporary string.

if((unencrypted[x][y] + rotation) > 96 && unencrypted[x][y] + rotation < 123)

{

tempString += char(unencrypted[x][y] + rotation);

}

else if(unencrypted[x][y] + rotation > 122 && unencrypted[x][y] + rotation < 149)

{

tempString += ((unencrypted[x][y] + rotation) - 26);

}

else

{

tempString += unencrypted[x][y];

}

}

//Push encrypted string into vector.

encrypted.push\_back(tempString);

}

}

void Crypt::encrypt(const vector<string> &unencrypted, vector<string> &encrypted)

{

string cryptogram = getKey("crypto.txt");

string tempString;

encrypted.clear();

//Iterate over each string in the vector.

for(int x = 0; x < unencrypted.size(); x++)

{

tempString.clear();

//Iterate over each charater of the current string.

for(int y = 0; y < unencrypted[x].size(); y++)

{

//Check if current character is a valid lowercase character.

//If it is, encrypt it and append it to the temporary string.

if(unencrypted[x][y] < 96 || unencrypted[x][y] > 122)

tempString += unencrypted[x][y];

else

{

//Find the correct replacement character from the cryptogram

//and replace the current character with it.

tempString += cryptogram[alphabet.find(unencrypted[x][y])];

}

}

//Push encrypted string into the vector.

encrypted.push\_back(tempString);

}

}

void Crypt::decrypt(const vector<string> &encrypted, vector<string> &unencrypted, int rotation)

{

unencrypted.clear();

string tempString ;

//Check if rotation is greater than the size of the entire alphabet.

if(rotation > 26)

rotation = rotation % 26;

//Iterate over each element in the vector.

for(int x = 0; x < encrypted.size(); x++)

{

tempString.clear();

//Iterate over character in the current element.

for(int y = 0; y < encrypted[x].size(); y++)

{

//Check if current character is a valid lowercase character.

//If it is, decrypt it and append it to the temporary string.

if((encrypted[x][y] - rotation) > 96 && encrypted[x][y] - rotation < 123)

{

tempString += char(encrypted[x][y] - rotation);

}

else if(encrypted[x][y] - rotation < 96 && encrypted[x][y] - rotation > 71)

{

tempString += ((encrypted[x][y] - rotation) + 26);

}

else

{

tempString += encrypted[x][y];

}

}

//Push decrypted string into vector.

unencrypted.push\_back(tempString);

}

}

void Crypt::decrypt(const vector<string> &encrypted, vector<string> &unencrypted)

{

string cryptogram = getKey("crypto.txt");

string tempString;

unencrypted.clear();

//Iterate over each element of the vector.

for(int x = 0; x < encrypted.size(); x++)

{

tempString.clear();

//Iterate over each character of the current element.

for(int y = 0; y < encrypted[x].size(); y++)

{

//Check if current character is a valid lowercase character.

//If it is, decrypt it and append it to the temporary string.

if(encrypted[x][y] < 96 || encrypted[x][y] > 122)

tempString += encrypted[x][y];

else

{

//Find the correct replacement character from the cryptogram.

//Replace the encrypted character with the corresponding

//character from the normal alphabet.

tempString += alphabet[cryptogram.find(encrypted[x][y])];

}

}

//Push decrypted string into the vector.

unencrypted.push\_back(tempString);

}

}

string getKey(string fileName)

{

string cryptogram;

fstream fin;

fin.open(fileName.c\_str());

fin >> cryptogram;

return cryptogram;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* a1.cpp

\* Driver file for encryption classes.

\*

\* @author Jeff Couch (couch009@cougars.csusm.edu

\* @version 1.1 (09.18.2014)

\*\*/

#include "FileIO.h"

#include "Crypt.h"

#include <iostream>

using namespace std;

int main()

{

vector<string> original;

vector<string> encrypted;

vector<string> decrypted;

int rotation;

bool flag;

char run;

char again = 'n';

do

{

int cryptType;

flag = 1;

do

{

cout << "=================================================\n"

<< "1. Rotation\n"

<< "2. Cryptogram\n"

<< "=================================================\n"

<< "Choose encryption type: ";

cin >> cryptType;

switch(cryptType)

{

case 1:

cout << "Input rotation amount: ";

cin >> rotation;

FileIO::readFile("original.txt", original);

Crypt::encrypt(original, encrypted, rotation);

FileIO::writeFile("encrypt01.txt", encrypted);

FileIO::readFile("encrypt01.txt", encrypted);

Crypt::decrypt(encrypted, decrypted, rotation);

FileIO::writeFile("decrypt01.txt", decrypted);

break;

case 2:

FileIO::readFile("original.txt", original);

Crypt::encrypt(original, encrypted);

FileIO::writeFile("encrypt02.txt", encrypted);

FileIO::readFile("encrypt02.txt", encrypted);

Crypt::decrypt(encrypted, decrypted);

FileIO::writeFile("decrypt02.txt", decrypted);

break;

default:

cout << "Invalid entry!\n";

flag = 0;

break;

}

}while(flag == 0);

cout << "Encrypt another (y/n)? ";

cin >> again;

}while(again == 'y');

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*original.txt\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

The recently approved Academic Blueprint adds new academic programs to the campus and guides planning for physical and curricular support for those programs. The additional programs will enhance the campus service to the region and provide new options and opportunities for students. Criminology and justice studies starts in fall 2003. We are also preparing for four new majors in fall 2004 - physical education and kinesiology, mass media, biochemistry, a master's in middle-level education and a doctorate in educational administration (in cooperation with SDSU and UCSD).

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*crypt.txt\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

wyijkcuvdpqlzhtgabmxefonrs