Assignment: Matrix Transposition Cipher

50 points.

(Nested Loop Processing, Row-Major Order, and Reversing It.)

Write a program which reads the standard input, stores text until it

encounters EOF, then encrypts the text using the Caesar Block Cipher.

SAMPLE RUN:

a.out

this is sample text suitable

for a simulation of a diplomatic

mission or a spy's instructions

^D

OUTPUT:

tpiaoosythltsnmisiieaioaoiostbmftnnnieluaiosssxeldcrtnstfaimarwasotpisulmurilspcr

----------------

Explanation:

It works like this:

Stage 1: User types:

this is sample text suitable

for a simulation of a diplomatic

mission or a spy's instructions

Stage 2: Remove all non-alpha chars // if (isalpha(c)) string s += c ;

Stage 3: Create a "square matrix" (an array of arrays of char with

enough spaces to store the message.

a. store is s.length() ?

b. calculate is the smallest perfect square that can hold it a message

of that length?

c. calculate: if that was a square, how long would a side be?

Stage 4: Allocate a two-dimensional array of arrays of char \_that\_

long on a side.

Stage 5:

step through string s and place the letters in the array

left to right, top to bottom

(Use Nested for() loops)

Step 6:

reverse the order of the for() loops, and pick off the

characters, top to bottom, left to right, output them, a character

at a time.

To decypher, do that very same thing with the encrypted msg.

In other words, the program both encyphers and decyphers,

symmetrically.

Example:

Stage 1:

this is sample text suitable

for a simulation of a diplomatic

mission or a spy's instructions

Stage 2:

thisissampletextsuitableforasimulationofadiplomaticmissionoraspysinstructions

Stage 3: The string's length() is 81.

Squares are: 2, 3, 9, 16, 25, 36, 49, 64, 81.

So 81 or 9 times 9, is the smallest table to hold the message.

(If the length of the message isn't a perfect square, we pick the

next-largest square number and pad it with random characters.)

(NEEDS TO BE SAID: don't store a table of squares. That wastes

space and cripples the program. CALCULATE squares using a for()

loop, and run it until your square is equal to or larger than

the length of the string of input text.)

Stage 4:

char table[9][9] ; (EXAMPLE: your code will differ, and the

literals will be replaced by a variable.)

Stage 5:

table holds:

thisistex

tsuitable

fortransl

ationinto

acipherto

demonstra

tethefunc

tioningof

thesystem

Stage 6:

Picking letters off from top to bottom, left to right, prints:

ttfaadttthsotceeihiuriimtoesitopohnsitrnhneiysaaiesfnstbnnrtugtelsttrnoexelooacfm

--------

By now, you understand arrays, and you understand for() loops, and you

understand nesting, so this should be a walk in the park for you.

(Now before everybody starts banging tin pans on the bars: this is a

lot more than the military cipher it appears. Yes, it's an historical

method for scrambling and descrambling military messages, but it's also

something we use frequently in programming. In fact, you will use it

later this semester for something completely different.)

STEPS TO A SOLUTION:

So: read your message into a large buffer or a string object. String

objects are more efficient and less dangerous, since they don't overflow

like CStrings.

Remove the spaces and punctuation(it's harder for the enemy to break

the code if you do).

Then count the chars in the message. (string::length())

Pick the first perfect square greater than the message length, allocate

an array of char that size.

Read the message into a square array of that size from left to right,

top to bottom.

IF THE MESSAGE DOESN'T FILL THE ARRAY COMPLETELY, INSERT RANDOM

LETTERS AT THE END. (WRITE A RANDOM PRINTABLE-CHARACTER GENERATOR

OR USE randchar.cpp.)

Learning outcomes:

Well, clearly, this teaches you how to process arrays with for() loops,

plus knowing how arrays are saved in the computer and how to access

their elements.

TEST: If you wrote a correct program, then this should output exactly

the letters that are input,(minus any spaces or punctuation):

$ cat file | caesarb | caesarb

All contents copyright (C) T.E.H., all rights reserved. None of this

code may be displayed or stored on any public access or private access

system without the written consent of the author. This means any and

all commercial cheating ("Homework Helping/Counselling") services.