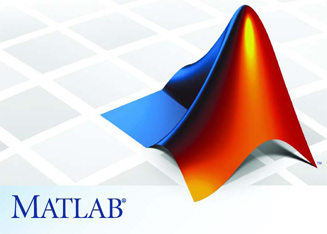
**THE ENIGMA CODE**

**EK 127 FALL 2012 PROJECT 1**



CARLTON DUFFETT  
SAM CHENEY  
LAB SECTION C6

PROFESSOR ATTAWAY

WE CERTIFY THAT ALL WORK PRESENTED HERE IS OUR OWN AND THAT NO OUTSIDE ASSISTANCE WAS RECEIVED IN THE DEVELOPMENT OF THIS CODE.

CARLTON DUFFETT \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DATE \_\_\_\_\_\_\_\_\_

SAM CHENEY \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DATE \_\_\_\_\_\_\_\_\_

HANDED IN TO PROJECT HAND IN FOLDER, SECTION C6. FOLDER TITLED: “CARLTON DUFFETT SAM CHENEY PROJECT 1”

main.m:

%This script simulates an Enigma machine with 3 rotors, including

%the ability to encode and decode a message, read a message from a file,

%and write a message to a file.

%

%Code developed by:

%Carlton Duffett

%Sam Cheney

%14 October 2012

clear

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%ENCRYPT OR DECRYPT?

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Prompt the user for encryption or decryption

cryptsel = input('Would you like to encrypt or decrypt (E/D)?\n','s');

if cryptsel == 'E' || cryptsel == 'e'

cryptsel = 1;

elseif cryptsel == 'D' || cryptsel == 'd'

cryptsel = 2;

else

fprintf('Error: Please make a valid selection.\n')

cryptsel = input('Would you like to encrypt or decrypt (E/D)?\n','s');

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%ROTOR SETTINGS

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%load rotors from file

load rotors.dat

%Ask user to select 3 rotors

fprintf('Please select the 3 rotors and starting letters.\n')

fprintf('You may choose from rotors I to V and a starting\n')

fprintf('position from 1 to 26.\n')

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%ROTOR I

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor I

rotoropt1 = input('Select rotor I: ');

%Error check

while rotoropt1 < 1 || rotoropt1 > 5 || round(rotoropt1) ~= rotoropt1

fprintf('Error: Please make a valid selection.\n')

rotoropt1 = input('Select rotor I: ');

end

%Store appropriate rotor

switch rotoropt1

case 1

rotorI = rotors(1,:);

case 2

rotorI = rotors(2,:);

case 3

rotorI = rotors(3,:);

case 4

rotorI = rotors(4,:);

case 5

rotorI = rotors(5,:);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor I starting position

rotorpos1 = input('Enter starting position for rotor I: ');

while rotorpos1 < 1 || rotorpos1 > 26 || round(rotorpos1) ~= rotorpos1

fprintf('Error - please enter a valid starting position.\n')

rotorpos1 = input('Enter starting position for rotor I: ');

end

%call rotorShift on rotor I to set rotor

for i = 1:(27-rotorpos1)

rotorI = rotorShift(rotorI);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%ROTOR II

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor II

rotoropt2 = input('Select rotor II: ');

%Error check

while rotoropt2 < 1 || rotoropt2 > 5 || rotoropt2 == rotoropt1 || round(rotoropt2) ~= rotoropt2

fprintf('Error: Please make a valid selection.\n')

rotoropt2 = input('Select rotor II: ');

end

%Store appropriate rotor

switch rotoropt2

case 1

rotorII = rotors(1,:);

case 2

rotorII = rotors(2,:);

case 3

rotorII = rotors(3,:);

case 4

rotorII = rotors(4,:);

case 5

rotorII = rotors(5,:);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor II starting position

rotorpos2 = input('Enter starting position for rotor II: ');

while rotorpos2 < 1 || rotorpos2 > 26 || round(rotorpos2) ~= rotorpos2

fprintf('Error - please enter a valid starting position.\n')

rotorpos2 = input('Enter starting position for rotor II: ');

end

%call rotorshift on rotor II to set rotor

for i = 1:(27-rotorpos2)

rotorII = rotorShift(rotorII);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%ROTOR III

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor III

rotoropt3 = input('Select rotor III: ');

%Error check

while rotoropt3 < 1 || rotoropt3 > 5 || rotoropt3 == rotoropt2 || rotoropt3 == rotoropt1 || round(rotoropt3) ~= rotoropt3

fprintf('Error: Please make a valid selection.\n')

rotoropt3 = input('Select rotor III: ');

end

%Store appropriate rotor

switch rotoropt3

case 1

rotorIII = rotors(1,:);

case 2

rotorIII = rotors(2,:);

case 3

rotorIII = rotors(3,:);

case 4

rotorIII = rotors(4,:);

case 5

rotorIII = rotors(5,:);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Select rotor III starting position

rotorpos3 = input('Enter starting position for rotor III: ');

while rotorpos3 < 1 || rotorpos3 > 26 || round(rotorpos3) ~= rotorpos3

fprintf('Error - please enter a valid starting position.\n')

rotorpos3 = input('Enter starting position for rotor III: ');

end

%call rotorshift on rotor III to set rotor

for i = 1:(27-rotorpos3)

rotorIII = rotorShift(rotorIII);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%INPUTS

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Ecryption:

if cryptsel == 1

%ask for plaintxt input

fprintf('Enter a message for encryption in all capital letters.\n')

plaintxt = input('Message: ','s');

%convert initial input

plaintxt = uint8(plaintxt);

%error check input

while any(plaintxt < 65) | any(plaintxt > 90)

fprintf('Error: Enter a message for encryption in all capital letters.\n')

fprintf('Exclude any spaces or underscores.\n')

plaintxt = input('Message: ','s');

end

%convert input

plaintxt = uint8(plaintxt);

end

%Decryption:

if cryptsel == 2

%load input

load ciphertext.dat

%error check input

while any(ciphertext < 65) | any(ciphertext > 90)

fprintf('Error: format of encrypted file is incompatible.\n')

fprintf('Ensure all values in file are ascii values between\n')

fprintf('65 and 90, arranged in a vector.\n')

end

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%THE ENIGMA CODE

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Encryption:

if cryptsel == 1

%Encrypt

enmsg = enigmaEncrypt(plaintxt,rotorI,rotorII,rotorIII);

end

%Decryption:

if cryptsel == 2

%Decrypt

demsg = enigmaDecrypt(ciphertext,rotorI,rotorII,rotorIII);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%OUTPUTS

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Encryption:

if cryptsel == 1

%Prompt user for output options

fprintf('Would you like your result saved as an ascii file,\n')

fprintf('printed in the command window, or both?\n')

outopt = input(' Save = 1\n Print = 2\n Both = 3\n Choice: ');

%Error check

while outopt < 1 || outopt > 3 || round(outopt) ~= outopt

fprintf('Error: please make a valid selection.\n')

outopt = input(' Save = 1\n Print = 2\n Both = 3\n Choice: ');

end

%Execute option selected

switch outopt

case 1

save ciphertext.dat enmsg -ascii

case 2

disp(char(enmsg))

case 3

save ciphertext.dat enmsg -ascii

disp(char(enmsg))

end

end

%Decryption:

if cryptsel == 2

%Prompt user for output options

fprintf('Would you like your result saved as an ascii file,\n')

fprintf('printed in the command window, or both?\n')

outopt = input(' Save = 1\n Print = 2\n Both = 3\n Choice: ');

%Error check

while outopt < 1 || outopt > 3 || round(outopt) ~= outopt

fprintf('Error: please make a valid selection.\n')

outopt = input(' Save = 1\n Print = 2\n Both = 3\n Choice: ');

end

%Execute option selected

switch outopt

case 1

save plaintext.dat demsg -ascii

case 2

disp(char(demsg))

case 3

save plaintext.dat demsg -ascii

disp(char(demsg))

end

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

enigmaEncrypt.m:

function enmsg = enigmaEncrypt(plaintxt,key1,key2,key3)

%This function encrypts a plaintext message letter by letter

%Preallocate output:

enmsg = zeros(1,length(plaintxt));

%Initialize character counter

counter = 0;

%loop through all letters in the message

for i = 1:length(plaintxt)

%encrypt with rotor I

enletter = subEncrypt(laintext(i),key1);

%encrypt with rotor II

enletter = subEncrypt(enletter,key2);

%encrypt with rotor III

enmsg(i) = subEncrypt(enletter,key3);

counter = counter + 1;

%shift rotors

if rem(counter,1) == 0

key1 = rotorShift(key1);

end

if rem(counter,26) == 0

key2 = rotorShift(key2);

end

if rem(counter,676) == 0

key3 = rotorShift(key3);

end

end

enigmaDecrypt.m:

function demsg = enigmaDecrypt(cyphertxt,key1,key2,key3)

%This function decrypts a cyphertext message letter by letter

%Preallocate output:

demsg = zeros(1,length(cyphertxt));

%Initialize character counter

counter = 0;

%loop through all letters in the message

for i = 1:length(cyphertxt)

%decrypt with rotor III

deletter = subDecrypt(cyphertxt(i),key3);

%decrypt with rotor II

deletter = subDecrypt(deletter,key2);

%decrypt with rotor I

demsg(i) = subDecrypt(deletter,key1);

counter = counter + 1;

%shift rotors

if rem(counter,1) == 0

key1 = rotorShift(key1);

end

if rem(counter,26) == 0

key2 = rotorShift(key2);

end

if rem(counter,676) == 0

key3 = rotorShift(key3);

end

end

rotorShift.m:

function newkey = rotorShift(key)

% This function shifts the input rotor key to the right by one letter.

%The last letter of the key becomes the first letter of the key.

%Preallocate output

newkey = zeros(1,26);

%Store last value of key

move = key(end);

%Shift key elements to right 1 space

for i = 1:25

newkey(i+1)=key(i);

%Place last element of key in first space

newkey(1) = move;

end

end

subEncrypt.m:

function cypherletter = subEncrypt(plainletter,key)

%This function converts a plaintext letter input to its corresponding

%cyphertext letter given a predetermined key

%encrypt:

cypherletter = key(plainletter-64);

end

subDecrypt.m:

function plainletter = subDecrypt(cypherletter,key)

%%This function converts a cypherletter letter input to its corresponding

%plaintext letter given a predetermined key

%Create alphabet for reference

alphabet = [65:90];

%decrypt:

plainletter = alphabet(find(key == cypherletter));

end

SHAKESPEARE’S “TO BE OR NOT TO BE” SPEECH GIVEN:

RINGSTELLUNG IV-22 V-4 I-12

TOBEORNOTTOBETHATISTHEQUESTIONWHETHERTISNOBLERINTHEMINDTOSUFFERTHESLINGSANDARROWSOFOUTRAGEOUSFORTUNEORTOTAKEARMSAGAINSTASEAOFTROUBLESANDBYOPPOSINGENDTHEMTODIETOSLEEPNOMOREANDBYASLEEPTOSAYWEENDTHEHEARTACHEANDTHETHOUSANDNATURALSHOCKSTHATFLESHISHEIRTOTISACONSUMMATIONDEVOUTLYTOBEWISHDTODIETOSLEEPTOSLEEPPERCHANCETODREAMAYTHERESTHERUBFORINTHATSLEEPOFDEATHWHATDREAMSMAYCOMEWHENWEHAVESHUFFLEDOFFTHISMORTALCOILMUSTGIVEUSPAUSETHERESTHERESPECTTHATMAKESCALAMITYOFSOLONGLIFEFORWHOWOULDBEARTHEWHIPSANDSCORNSOFTIMETHEOPPRESSORSWRONGTHEPROUDMANSCONTUMELYTHEPANGSOFDESPISEDLOVETHELAWSDELAYTHEINSOLENCEOFOFFICEANDTHESPURNSTHATPATIENTMERITOFTHEUNWORTHYTAKESWHENHEHIMSELFMIGHTHISQUIETUSMAKEWITHABAREBODKINWHOWOULDFARDELSBEARTOGRUNTANDSWEATUNDERAWEARYLIFEBUTTHATTHEDREADOFSOMETHINGAFTERDEATHTHEUNDISCOVERDCOUNTRYFROMWHOSEBOURNNOTRAVELLERRETURNSPUZZLESTHEWILLANDMAKESUSRATHERBEARTHOSEILLSWEHAVETHANFLYTOOTHERSTHATWEKNOWNOTOFTHUSCONSCIENCEDOESMAKECOWARDSOFUSALLANDTHUSTHENATIVEHUEOFRESOLUTIONISSICKLIEDOERWITHTHEPALECASTOFTHOUGHTANDENTERPRISESOFGREATPITHANDMOMENTWITHTHISREGARDTHEIRCURRENTSTURNAWRYANDLOSETHENAMEOFACTIONSOFTYOUNOWTHEFAIROPHELIANYMPHINTHYORISONSBEALLMYSINSREMEMBERD

ENCODED TO READ:



DECODED decryptMe.dat GIVEN:

RINGSTELLUNG III-18 V-21 IV-13

ALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOYALLWORKANDNOPLAYMAKESJAKEADULLBOY