PROJECT 2:
Encoding Assistants

CSE100/INFO100 Fluency with Information Technology

Spring 2002

Introduction

Codes and encoding are an important part of computing. Often the codes used at one level are completely different from those at the next level. For example, pages of raw HTML seldom resemble the page as rendered by a browser.

Of course, codes are not unique to computing. The human genetic code is a biological example. The telegraph introduced perhaps the first widely used binary representation of writing: the Morse code. Sending a telegram actually involved at least four levels of coding. First, the verbal message had to be written down (coded into letters). Then the message was translated into upper case only, and all punctuation was replaced with special uppercase words. The result of that encoding step was then ready to undergo Morse coding. Finally, the abstract Morse coding itself was mechanically transformed into electrical impulses for transmission to a remote location. The first stage might be carried out by the end user or the operator; the second stage was a mental one, performed by the operator as he or she manually performed the third step. The last step was carried out mechanically by the telegraph equipment.

Sometimes people are better at coding tasks, and sometimes computers are. Sometimes the computer's role is simply to assist the human. For example, despite decades of research, translating text from one language to another is very difficult for a computer. For high-quality results, the best systems involve both a human operator and computer-based assistance. There are also some coding tasks so difficult that neither people nor computers can do them effectively.

In this project, we'll explore two environments in which a computer might assist a human in a coding task. You are invited to think of other applications and how they do or not fit this model of human-computer interaction.

Objectives

- To design and implement several VB applications that will transform and encode text messages
- To develop understanding of various ways to represent information

Helpful Reading Resources for both parts of Project 2

- Chapter 6 of *Computer Programming Fundamentals with Applications in Visual Basic 6.0*
- p. 265-269 (top of page) in Chapter 10 of *Computer Programming Fundamentals with Applications in Visual Basic 6.0*
- Chapter 6.2 and 6.3: Built-in Functions AND Writing Functions and Subroutines from *The Visual Basic Coach*. This is on e-Reserve at the library: https://eres.lib.washington.edu/coursepage.asp?cid=1113&page=01

Part 1: Telegraph Operator's Assistant
Due Friday, May 10th, 9:00 PM through eSubmit

Read the instructions given on Project TurnIn page of our website. With each project piece you will also submit a paper copy of your code and form. The paper copies will be due in the lecture following the due date.

What to do

Your goal is to create an application that will transform messages given to the telegraph assistant to the upper case encoding needed to convert them to Morse Code and send them across the telegraph wires.

The screen capture of the form shown below is a sample of how your application interface should look. You may choose any color scheme and design you wish-but the basic objects shown must be included.

Program Logic

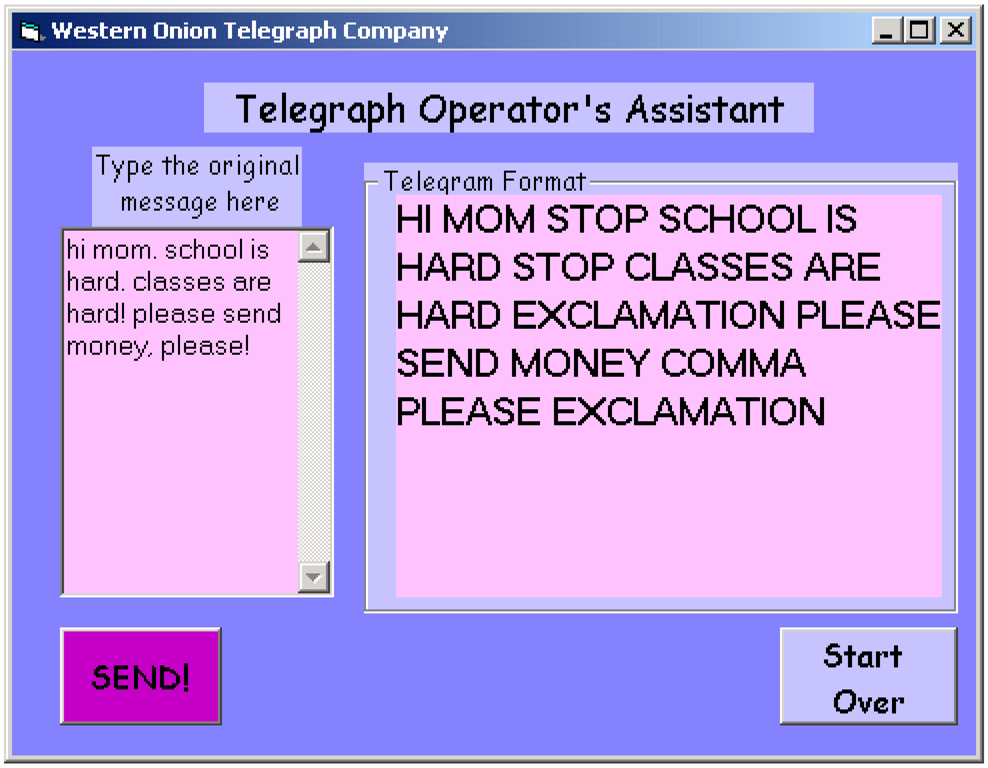
A user enters a message into a text box on the left. As they enter the keyboard letters, a transformed version of the message appears in the label on the right, ***letter by letter.***

When the message is complete, the Send button is pressed, producing screen 2 (at a minimum). The start over button should reset the screen so another message can be encoded. Try this sample executable of the program. (Runs on Windows only) Make your program have at least the same functionality as the sample.

Your application should convert all letters in the alphabet to uppercase AND convert the common grammar notations to uppercase words when they are entered.

.	PERIOD
,	COMMA
:	COLON
;	SEMICOLON
?	QUESTION
!	EXCLAMATION

Figure 1: Interface with encoding of text from left side to right



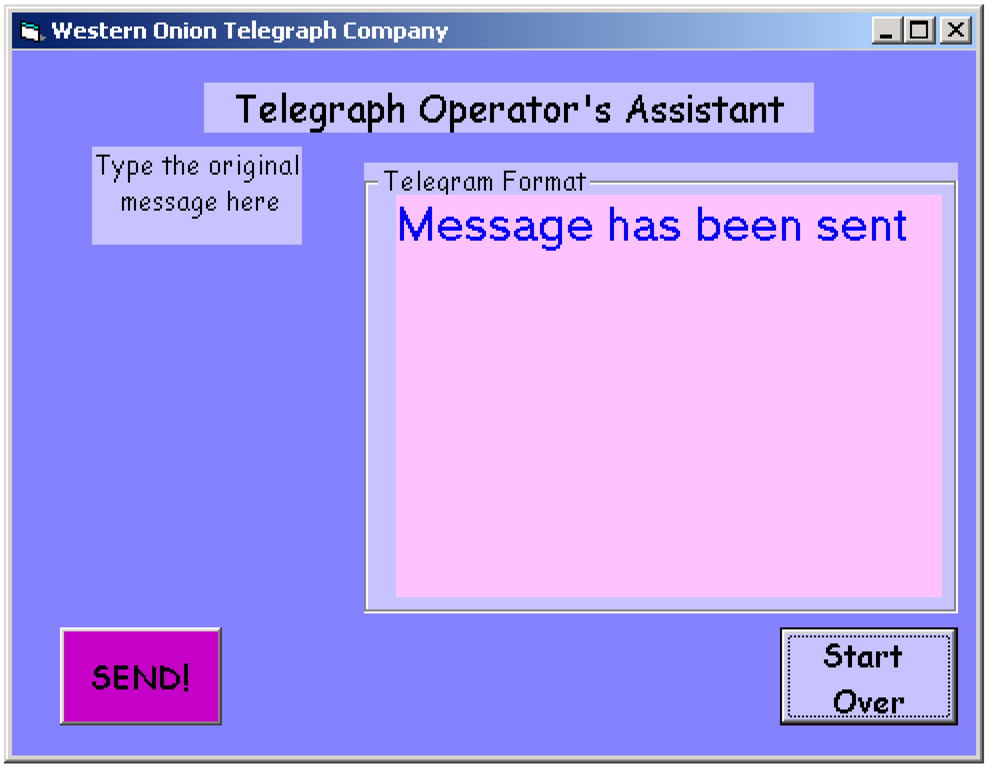
Western Onion Telegraph Company

Telegraph Operator's Assistant

hi mom. school is hard. classes are hard! please send money, please!

HI MOM STOP SCHOOL IS HARD STOP CLASSES ARE HARD EXCLAMATION PLEASE SEND MONEY COMMA PLEASE EXCLAMATION

Figure 2: Interface after Send button is pressed



Western Onion Telegraph Company

Telegraph Operator's Assistant

Message has been sent

Before sitting down to the computer, open your lab note book and plan the implementation of your project piece by walking through the logic. Answer the following questions for yourself:

- What objects need to be on my form to complete the interface?
- What will the names be so I can read through my code and understand what an object is doing?
- What event procedures will be needed to complete the encoding of my message?
- How will I determine when to use simple letter transformation versus substituting an entire word?
- What should the flow of the program be? (Look at the sample executable for suggestions)

Part 2: Cryptographer's Assistant
Due Wednesday, May 22nd, MIDNIGHT through eSubmit

Read the instructions given on Project TurnIn page of our website. With each project piece you will also submit a paper copy of your code and form. The paper copies of your code, form and writing component will be due in the next lecture following the due date-Friday, May 24th.

In cryptography, the goal of coding is to disguise the message so that only its intended recipients can decode it. Crypto codes are very ancient. In recent years, cryptography has used advanced mathematics and the power of computers to create seemingly unbreakable codes.

In our project, we will use a long-known technique of "substitution". Some versions are called a "Caesar Cipher" because Julius Caesar is said to have used this technique.
The idea is this: each symbol in the original text is replaced by another letter or symbol. The table of substitutions is called the "code book." The original, uncoded message is referred to as "clear text."

For this part of the project, we will translate all characters to upper case first, and do no substitution of non-letter symbols. That is, spaces, number, punctuation, etc. will be passed through without change. This would have been fine for Julius Caesar. In his day, words were generally written all in upper case, without punctuation, and FREQUENTLYWITHOUTSPACESBETWEENTHEWORDS.

What to do

Your task in Part 2 of the assignment is to build a VB application to help Caesar cipher. The work you did for part 1 will be helpful for part 2. Use much of the same logic in building the Cryptographer's Assistant as you did for the telegraph operator's assistant. Follow the program logic and step below to help you arrive at the expected outcome. There is also a written component to this project piece. See the description just after the sample Cryptographer's Assistant Interface

Program Logic

Your application will allow someone to encode a message by entering clear text on the left and seeing the encrypted text on the right. This is a carry over from Part 1. However, you will not encode additional grammar notations as you did in Part 1.

At any point during the encryption, the user should be able to change the codebook letter by letter and re-encrypt the message with the simple push of a button.

Requirements

Your solution should have the following for mostly full credit:

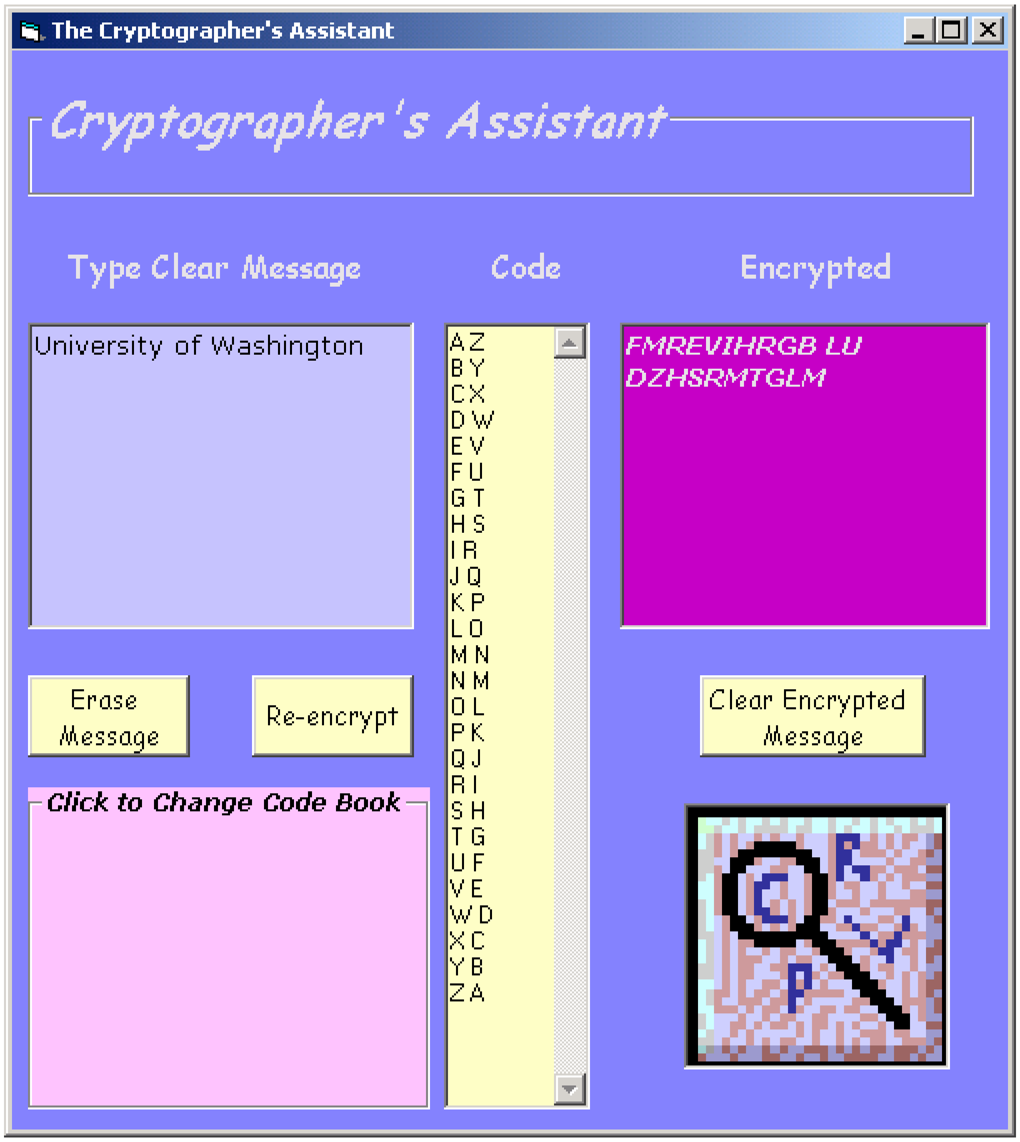
1. Suitable controls (objects), arranged for convenience and ease of use. Take into consideration how usable your interface is as you design it. Make sure you rename all of them. Points off for using the default names.
2. A valid codebook that maps each character to some other symbol - Take care not to reuse symbols.
3. In your VB program, maintain the codebook as a Collection, whose keys are the uppercase letters "A", "B", ... "Z" (Lab 11/12 will go over Collections)
4. Show the full codebook on the form (use a text box with scroll bars)
5. Allow "Caesar" to enter clear text messages and have the Assistant correctly encode and display the results:
    A. All letters, upper or lower case, should be translated according to the code book
    B. All other symbols should be placed in the output message without change.
6. Allow Caesar to clear the input message and start over from scratch.
7. Submit your write-up, in class on May 22nd.

To get the last portion of credit, you must also:

8. Provide a way for Caesar to change the codebook, letter by letter.
9. Provide a way to re-encode the message after the codebook has been changed.

Here is one sample version to play with. It meets all of the basic requirements except one: its codebook is not valid, because each letter is mapped to itself.

Figure 3: Sample Interface After User changes codebook


The Cryptographer's Assistant
Cryptographer's Assistant
Type Clear Message
Code
Encrypted
Erase Message
Re-encrypt
Clear Encrypted Message
Click to Change Code Book

University of Washington

A	Z
B	Y
C	X
D	W
E	V
F	U
G	T
H	S
I	R
J	Q
K	P
L	O
M	N
N	M
O	L
P	K
Q	J
R	I
S	H
T	G
U	F
V	E
W	D
X	C
Y	B
Z	A

FMREVIHRGB LU
DZHSRMTGLM

Writing Component: Code Breaker's Assistant

What To Do

We've reached the final stage. The message has been encrypted and sent. At the other end, the receiver decodes the message, using the codebook. In order for this process to be secure, of course, the codebook has to be transmitted to the receiver by some separate and very secure method. You can imagine programming a Decoder's Assistant, which would be very much like the Cryptographer's Assistant, perhaps even simpler. But this is not your task.

Imagine this scenario: on the way from point A to point B, secret agent 0001 (“triple-O-One”) intercepts the message sent from your application. Now he or she just needs to decode it to take over the World! But there is a slight problem: no codebook.

That's where you come in. You will try to design a Code Breaker's Assistant. YOU DO NOT NEED TO CREATE THE APPLICATION!!! Your design description will help the code breaker to create the application to decrypt the message. Explain your design in as much detail as possible. You might even want to sketch out an interface so you have an object to focus on.

Start your work under with the following premise:

You are given the secret message that was intercepted by Agent 001. From this message you need to describe an application that would break the code.

Imagine this is the intercepted message:

RU BLF WVXLWV 3SRH 2VHHZTV DR3SLF3 FHV LU Z KILTIZ2, 3SVM BLF SZEV 2ZH3VIVW 3SV ZI3 LU OZMTFZTV ZMW TIZ22VI 2ZMRKFOZ3RLM ZH Z DZB 3L OLLP ULI 3SV2VH ZMW.

Questions to consider:

- Ask yourself, if *you* were handed the message (say your life depended on it), would this be enough?
- What kind of process would you, as a person, go through to find the secret meaning?
- Is there any process you could follow without a program to try and decrypt this message? Explain it in detail
- What parts of that process could a computer help you with?
- How important is the user interface in this process? (What is it about an interface that makes the act of using the program simple or intuitive?
- What makes a good user interface?

In thinking about the questions above, follow two guidelines:

- Do *not* take into account your current skill as a programmer

- *Do* take into account what you have learned about computers: what kinds of things they can do, what's easy, what's hard, etc.

The "Questions to Consider" questions should be answered as the first part of your design description.

The second part of the design should be a list of features which, if added to a Code Breaker's Assistant program by a skilled VB programmer, would help the code breaker do a better job.

For any feature you propose, you need to have an idea of what steps are involved in implementing it, even if you couldn't currently write the program. For example, don't just say, "The Assistant would crack the code automatically." Why not? Because nothing a computer does is automatic. It needs an algorithm to tell it HOW to anything. So you have to give an algorithm for each of your suggestions. The algorithm can be quite informally given, as long as it seems plausible to a computer-knowledgeable person (such as your TA!). The features you think up don't necessarily have to be complex or impressive. They might include little things, like style or format... if it would assist the code breaker.

Now, after thinking hard about this and coming up with some ideas, make two lists:

- A-list: Features that you think you could implement, given your VB knowledge, or just a bit beyond your current knowledge. Or rather, features you think the instructor would expect you to be able to do (or just a bit beyond).
- B-list: Features that are clearly beyond your current programming experience, but which you feel could in principle be implemented from your description and instructions.

Your final product for part 2 will include:

- The Cryptographer's Assistant application (form, project and executable files submitted electronically)
- A print out of the code and the form used
- A description of how you would go about decrypting the message given WITHOUT any program provided. (Bonus of 2 points if you DO decrypt it and include the message!)

- A description of an application that would decrypt the message. This will include some features from your A list, at least.
- A write up and description of the features for your A and B list.