Information Technology PAT

Connor Hess

Paarl Boys High

Information Technology

Contents

[Scenario 2](#_Toc56016941)

[User Requirements 3](#_Toc56016942)

[Navigation/Description of flow diagram 4](#_Toc56016943)

[GUI design 5](#_Toc56016944)

[frmMain\_Encrypt 5](#_Toc56016945)

[frmMain\_Encrypt data structures 5](#_Toc56016946)

[Strings: 5](#_Toc56016947)

[Integers: 5](#_Toc56016948)

[frmMain\_Encrypt IPO table 5](#_Toc56016949)

[frmMain\_Decrypt 6](#_Toc56016950)

[frmMain\_Decrypt data structures 6](#_Toc56016951)

[Strings: 6](#_Toc56016952)

[Integers: 6](#_Toc56016953)

[frmMain\_Encrypt IPO table 6](#_Toc56016954)

# Scenario

An agency needs an application to add an added layer of security to the text communications between their branches. They need it to be a unique encryption that is not used by many people. The string must be unrecognizable and needs to be able to be decrypted without the use of the program if needed. It needs to be a lightweight encryption so that it can be put onto smaller, less powerful devices to be used it the field. The program must not connect to any servers and must not require internet to function, I need to be simple and easy to use.

To solve this problem, I will make an application to encrypt and decrypt strings that can be sent or written and mailed without other people being able to decipher it. It will use a simple ascii shift containing 2 conditions that can be changed to make it more difficult to decrypt.

# User Requirements

In this program I will have 2 users. One will be the “sender” who needs to encrypt his message and the other the “receiver” who needs to read that message.

|  |  |  |  |
| --- | --- | --- | --- |
| As a … | I would like to … | So that I can … | **Acceptance test** |
| Sender | Encrypt the message | Send the message so that only the other person can read it | **Encrypt data** –  Data not recognisable |
| Receiver | Decrypt the message | Read the received message | **Decrypt data** –  Data is recognisable |

# Navigation/Description of flow diagram

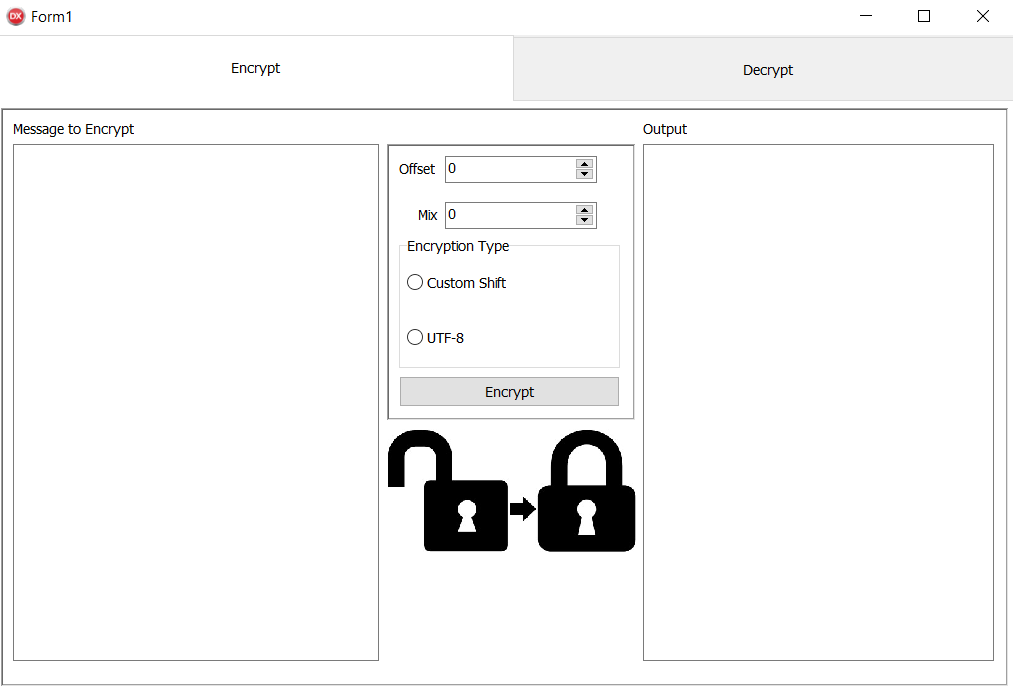
Encrypt Tab

Decrypt Tab

Main Form

# GUI design

## frmMain\_Encrypt



## frmMain\_Encrypt data structures

variables that I will use

### Strings:

sInput\_enc – the message input.

sOutput\_enc – the formatted message.

### Integers:

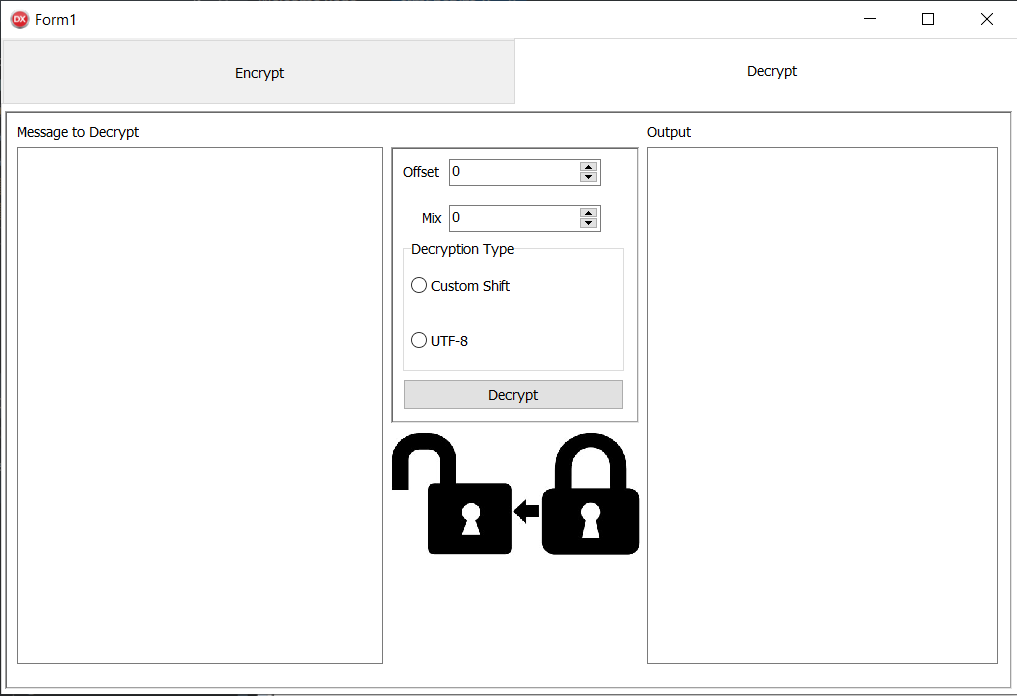
iOffset\_enc – the amount added to the ascii value.

iMix\_enc – a number by which to devide the number.

## frmMain\_Encrypt IPO table

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| Keyboard input   * edtInput   The input needs to be in the following format:  *Any text* | * presence check | none |
| Component inputs   * Button – Encrypt * SpinEdit – Encrypt Mix * SpinEdit – Encrypt Offset * RadioGroup – type of Encryption | Based on the type of Encryption, I will take the the edtInput and convert it to ASCII values. I will take the ascii value and add the offset. Then I will connect the ascii values into one string and then devide by the mix  Start  sInput\_dec <- edtInput.text  loop length of sInput\_dec  mem\_output\_encrypt.text <- (ord(sInput\_dec[enumerated value]) \* iMix) + iOffset  Stop | Encrypted string – mem\_output\_encrypt |

## frmMain\_Decrypt



## frmMain\_Decrypt data structures

variables that I will use

### Strings:

sInput\_dec – the message input.

sOutput\_dec – the formatted message.

### Integers:

iOffset\_dec – the amount deducted from the ascii value.

iMix\_dec – a number by which to multiply the ascii values.

## frmMain\_Encrypt IPO table

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| Keyboard input   * edtInput   The input needs to be in the following format:  *Any text* | * presence check | nones |
| Component inputs   * Button – Decrypt * SpinEdit – Decrypt Mix * SpinEdit – Decrypt Offset * RadioGroup – type of Decryption | Based on the type of Decryption, I will take the the edtInput and multiply by the mix. Then I will separate the values. Then I will subtract the offset. And convert it back into a string  Start  sInput\_dec <- edtInput.text  loop length of sInput\_dec  mem\_output\_dec.text <- (ord(sInput\_dec[enumerated value]) - iOffset) / iMix  Stop | Decrypted string – mem\_output\_decrypt |