

111E000003 電子電路自動化通論

Base64 編碼轉換

使用 LabWindows

第四組

日電子四甲 U0822039 楊淳佑

日資工三甲 U0924043 陳皇任

文本目錄

文本目錄.....	1
圖表目錄.....	1
壹、動機與目的.....	2
貳、內容.....	3
一、版面配置.....	3
二、程式設計.....	3
三、執行畫面（按照測試步驟）.....	6
參、討論.....	8
一、版面配置與功能說明.....	8
二、編碼時自動對齊.....	8
肆、心得.....	8

圖表目錄

圖表 1 主程式版面配置.....	3
圖表 2 資料設定集成函式，在各項操作中被重複使用.....	3
圖表 3 版面配置紅色序號 1，單元編碼按鈕.....	4
圖表 4 版面配置紅色序號 2，單元解碼按鈕.....	4
圖表 5 版面配置紅色序號 3，全文編碼按鈕.....	5
圖表 6 版面配置紅色序號 4，全文解碼按鈕.....	5
圖表 7 執行畫面：第零步：程式啟動畫面.....	6
圖表 8 執行畫面：第一步：輸入一段測試文字（英數等半形字元）.....	6
圖表 9 執行畫面：第二步：按下全文編碼按鈕進行編碼.....	7
圖表 10 執行畫面：第三步：按下全文解碼按鈕進行解碼.....	7

壹、動機與目的

若需將目標文字直接以字串的方式在程式中傳遞，容易因目標文字中，含有一些特殊符號，進而導致該字串結構被破壞，影響程式之執行，甚至受到注入攻擊（Injection Attack）。因此為了不讓程式易受到目標文字的破壞，我們想將目標文字之內容，轉換為一些更加安全的字符。

我們選擇將原始內容轉換為 Base64，將原始內容改以較安全之 64 個字符儲存，同時降低轉換時的複雜度，亦可使該轉換後之內容，可以較高效之位元空間、安全的字符傳遞。我們將以 CVI 製作出能夠在 ACSIIcode 以及 Base64 之間來回轉換之程式，並讓使用者可以瞭解其運作機制。

貳、內容

一、版面配置



圖表 1 主程式版面配置

二、程式設計



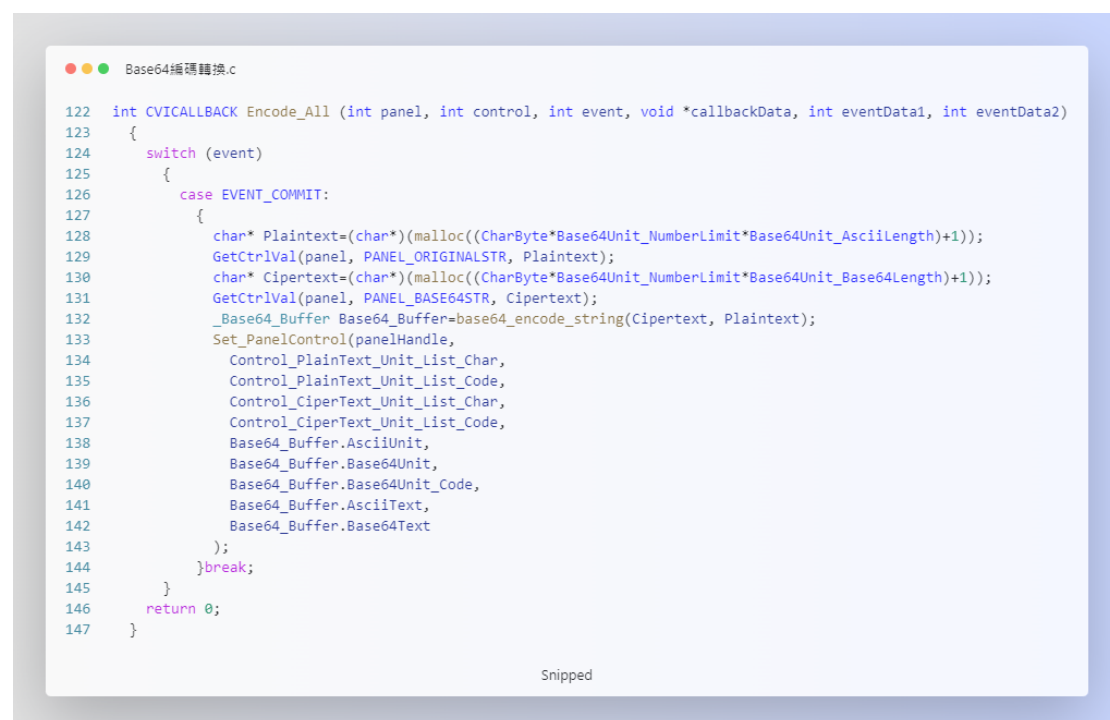
圖表 2 資料設定集成函式，在各項操作中被重複使用



圖表 3 版面配置紅色序號 1，單元編碼按鈕



圖表 4 版面配置紅色序號 2，單元解碼按鈕



```
Base64編碼轉換.c

122 int CVICALLBACK Encode_All (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
123 {
124     switch (event)
125     {
126     case EVENT_COMMIT:
127     {
128         char* Plaintext=(char*)(malloc((CharByte*Base64Unit_NumberLimit*Base64Unit_AsciiLength)+1));
129         GetCtrlVal(panel, PANEL_ORIGINALSTR, Plaintext);
130         char* Cipertext=(char*)(malloc((CharByte*Base64Unit_NumberLimit*Base64Unit_Base64Length)+1));
131         GetCtrlVal(panel, PANEL_BASE64STR, Cipertext);
132         _Base64_Buffer Base64_Buffer=base64_encode_string(Cipertext, Plaintext);
133         Set_PanelControl(panelHandle,
134             Control_PlainText_Unit_List_Char,
135             Control_PlainText_Unit_List_Code,
136             Control_CiperText_Unit_List_Char,
137             Control_CiperText_Unit_List_Code,
138             Base64_Buffer.AsciiUnit,
139             Base64_Buffer.Base64Unit,
140             Base64_Buffer.Base64Unit_Code,
141             Base64_Buffer.AsciiText,
142             Base64_Buffer.Base64Text
143         );
144     }break;
145     }
146     return 0;
147 }
```

Snipped

圖表 5 版面配置紅色序號3，全文編碼按鈕



```
Base64編碼轉換.c

148 int CVICALLBACK Decode_All (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
149 {
150     switch (event)
151     {
152     case EVENT_COMMIT:
153     {
154         char* Plaintext=(char*)(malloc((CharByte*Base64Unit_NumberLimit*Base64Unit_AsciiLength)+1));
155         GetCtrlVal(panel, PANEL_ORIGINALSTR, Plaintext);
156         char* Cipertext=(char*)(malloc((CharByte*Base64Unit_NumberLimit*Base64Unit_Base64Length)+1));
157         GetCtrlVal(panel, PANEL_BASE64STR, Cipertext);
158         _Base64_Buffer Base64_Buffer=base64_decode_string(Plaintext, Cipertext);
159         Set_PanelControl(panelHandle,
160             Control_PlainText_Unit_List_Char,
161             Control_PlainText_Unit_List_Code,
162             Control_CiperText_Unit_List_Char,
163             Control_CiperText_Unit_List_Code,
164             Base64_Buffer.AsciiUnit,
165             Base64_Buffer.Base64Unit,
166             Base64_Buffer.Base64Unit_Code,
167             Base64_Buffer.AsciiText,
168             Base64_Buffer.Base64Text
169         );
170     }break;
171     }
172     return 0;
173 }
```

Snipped

圖表 6 版面配置紅色序號4，全文解碼按鈕

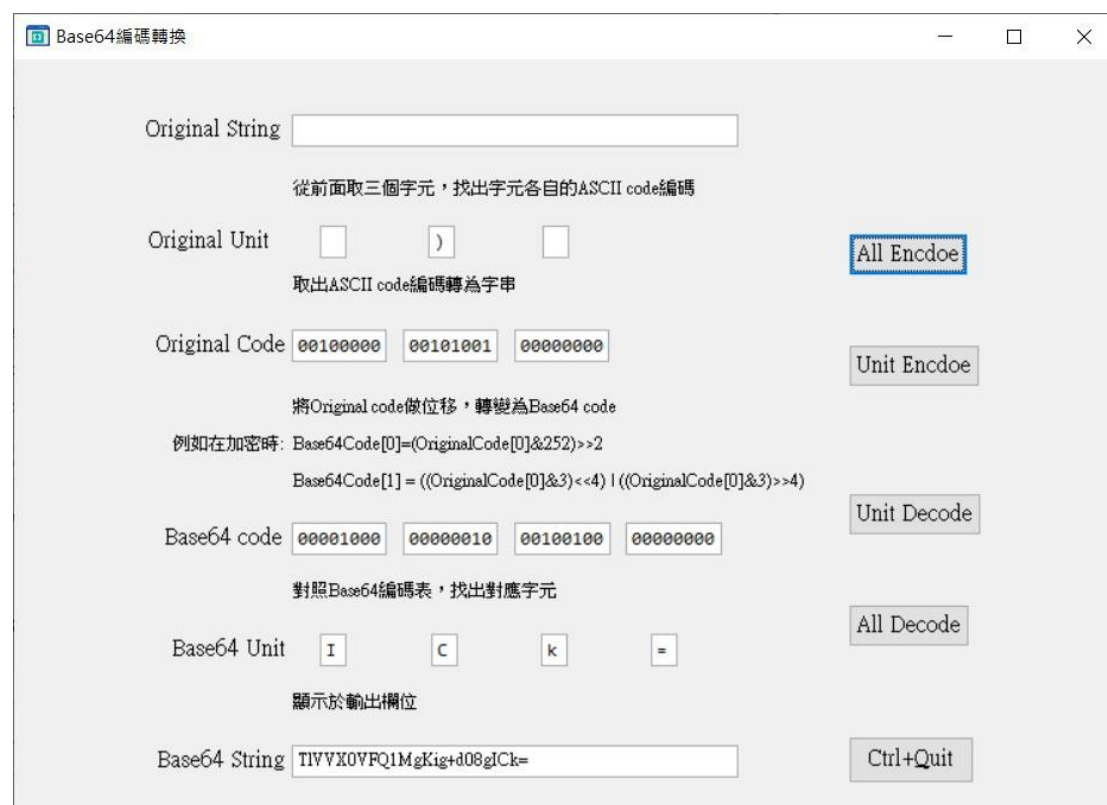
三、執行畫面（按照測試步驟）

The screenshot shows the 'Base64 編碼轉換' application window. It contains several input fields and buttons for encoding and decoding Base64 strings. The 'Original String' field is empty. Below it, a text label reads '從前面取三個字元，找出字元各自的ASCII code編碼'. The 'Original Unit' field has three empty boxes. To its right is an 'All Encode' button. Below this, a text label reads '取出ASCII code編碼轉為字串'. The 'Original Code' field has three empty boxes. To its right is a 'Unit Encode' button. Below this, a text label reads '將Original code做位移，轉變為Base64 code'. Two lines of code are displayed: $\text{Base64Code}[0] = (\text{OriginalCode}[0] \& 252) \gg 2$ and $\text{Base64Code}[1] = ((\text{OriginalCode}[0] \& 3) \ll 4) | ((\text{OriginalCode}[1] \& 3) \gg 4)$. The 'Base64 code' field has four empty boxes. To its right is a 'Unit Decode' button. Below this, a text label reads '對照Base64編碼表，找出對應字元'. The 'Base64 Unit' field has four empty boxes. To its right is an 'All Decode' button. Below this, a text label reads '顯示於輸出欄位'. The 'Base64 String' field is empty. To its right is a 'Ctrl+Quit' button.

圖表 7 執行畫面：第零步：程式啟動畫面

The screenshot shows the 'Base64 編碼轉換' application window with the 'Original String' field containing the text 'NUU_EECS *(>wO)'. The rest of the interface is the same as in the previous screenshot, with the 'Original Unit' field having three empty boxes, 'Original Code' having three empty boxes, 'Base64 code' having four empty boxes, 'Base64 Unit' having four empty boxes, and 'Base64 String' being empty.

圖表 8 執行畫面：第一步：輸入一段測試文字（英數等半形字元）



The screenshot shows the 'Base64編碼轉換' window at the second step of the encoding process. The 'Original String' field is empty. The 'Original Unit' field shows a space character. The 'Original Code' field shows the binary representation of the space character: 00100000 00101001 00000000. The 'Base64 code' field shows the binary representation of the Base64 code: 00001000 00000010 00100100 00000000. The 'Base64 Unit' field shows the characters 'I', 'C', 'k', and '='. The 'Base64 String' field shows the encoded string: TIVVX0VFQ1MgKig+d08gICk=.

Original String

從前面取三個字元，找出字元各自的ASCII code編碼

Original Unit)

取出ASCII code編碼轉為字串

Original Code 00100000 00101001 00000000

將Original code做位移，轉變為Base64 code

例如在加密時: $\text{Base64Code}[0] = (\text{OriginalCode}[0] \& 252) \gg 2$

$\text{Base64Code}[1] = ((\text{OriginalCode}[0] \& 3) \ll 4) | ((\text{OriginalCode}[1] \& 3) \gg 4)$

Base64 code 00001000 00000010 00100100 00000000

對照Base64編碼表，找出對應字元

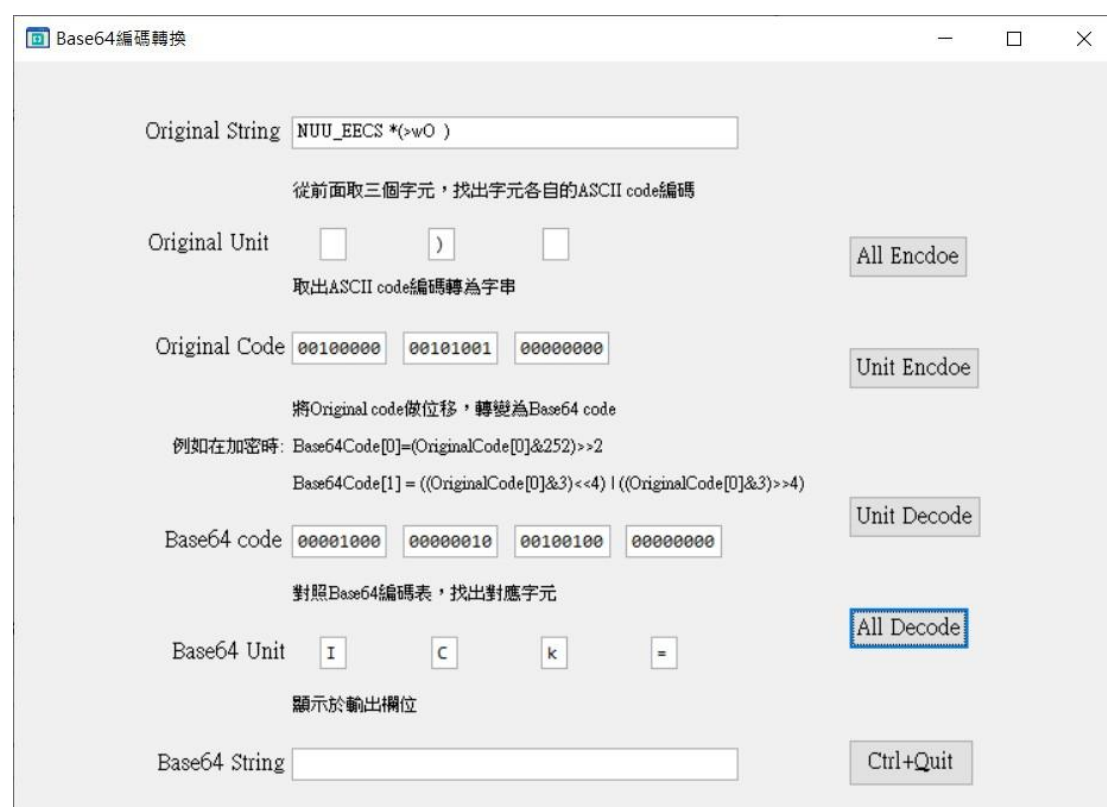
Base64 Unit I C k =

顯示於輸出欄位

Base64 String TIVVX0VFQ1MgKig+d08gICk=

Buttons: All Encode, Unit Encode, Unit Decode, All Decode, Ctrl+Quit

圖表 9 執行畫面：第二步：按下全文編碼按鈕進行編碼



The screenshot shows the 'Base64編碼轉換' window at the third step of the decoding process. The 'Original String' field contains the encoded string: NUU_EECS *(>wO). The 'Original Unit' field shows a space character. The 'Original Code' field shows the binary representation of the space character: 00100000 00101001 00000000. The 'Base64 code' field shows the binary representation of the Base64 code: 00001000 00000010 00100100 00000000. The 'Base64 Unit' field shows the characters 'I', 'C', 'k', and '='. The 'Base64 String' field is empty.

Original String NUU_EECS *(>wO)

從前面取三個字元，找出字元各自的ASCII code編碼

Original Unit)

取出ASCII code編碼轉為字串

Original Code 00100000 00101001 00000000

將Original code做位移，轉變為Base64 code

例如在加密時: $\text{Base64Code}[0] = (\text{OriginalCode}[0] \& 252) \gg 2$

$\text{Base64Code}[1] = ((\text{OriginalCode}[0] \& 3) \ll 4) | ((\text{OriginalCode}[1] \& 3) \gg 4)$

Base64 code 00001000 00000010 00100100 00000000

對照Base64編碼表，找出對應字元

Base64 Unit I C k =

顯示於輸出欄位

Base64 String

Buttons: All Encode, Unit Encode, Unit Decode, All Decode, Ctrl+Quit

圖表 10 執行畫面：第三步：按下全文解碼按鈕進行解碼

參、討論

一、版面配置與功能說明

一) 問題

這個程式不像遊戲，只需要讓使用者理解自身需要怎麼操作，而是需要使用者也理解內部的運行方式，所以需要多加說明其各個步驟的作用。

二) 解方

試著讓非本專業人事理解流程，再與其討論應如何設計版面配置，並加註許多的解釋來說明個步驟的用處，使他人較易理解。

二、編碼時自動對齊

一) 問題

在編碼的過程中通常以一個單元來進行編碼，若被編碼之字元長度不足，所生成編碼長度亦會不足。通常會將所生成之編碼長度以對其字元來補齊應有的長度。

二) 解方未來或許可以

比較被編碼的字元長度與生成字元長度是否一樣，若一樣則代表被編碼的字元有湊滿 3 個字元，照常轉換為 base64；若是小於則代表字串長度不足 3 個字元，則需要以 '=' 來填補缺少的部分。何時該使用對其字元之敘述式，最後設計方案如下：

```
(((((int)((AsciiBit*alignLength)-(Base64Bit*index)))>0)?(1):(0))
```

- `alignLength`：被編碼之字元長度
- `index`：生成編碼之字元長度
- `AsciiBit`：被編碼之單元長度
- `Base64Bit`：生成編碼之單元長度

肆、心得

在我們的課程中，關於視窗程式通常都是使用 C#、java 或 Python 來撰寫。透過這個課程，我們了解到 CVI(LabWindows)略為不同的架構：使用 Callback，而非 Listener。以及使用非物件化的程式撰寫。

未來或許可以在功能上增加針對檔案串流的編碼轉換，讓使用者在使用程式時可以更加的方便。