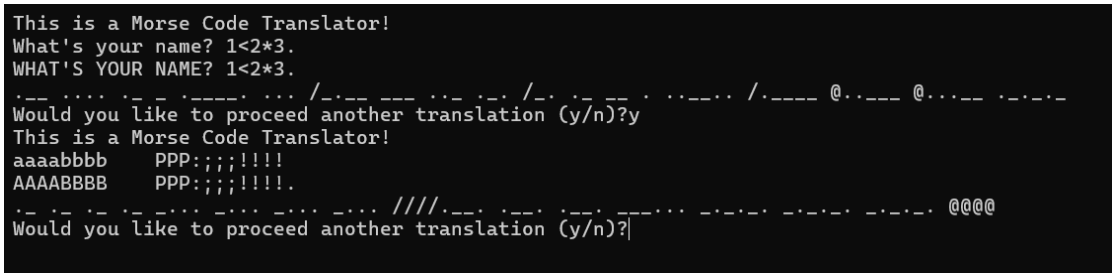

字串，array 設定初始化

```
1       ; IDIV Examples            (Idiv.asm)
2
3       ; This program shows examples of various IDIV formats.
4
5       INCLUDE Irvine32.inc
6
7       BUFMAX = 128
8       .data
9
10      Welc    BYTE "This is a Morse Code Translator! ",0
11      Input   BYTE "Please enter your string:",0
12      Output  BYTE "Morse Code:",0
13      Again   BYTE "Would you like to proceed another translation (y/n)?",0
14      prompt  DWORD 128 DUP(?)
15      upper   DWORD 128 DUP(?)
16      morse   DWORD 128 DUP(?)
17      bufSize DWORD  ?
```

A~Z 初始化，將每個字都設定為 8 個 byte

```
19      Alphabet    BYTE "._",0,0,0,0,0,0
20                  BYTE "_...",0,0,0,0
21                  BYTE "_._.",0,0,0,0
22                  BYTE "_..",0,0,0,0,0
23                  BYTE ".",0,0,0,0,0,0,0
24                  BYTE ".._.",0,0,0,0
25                  BYTE "__.",0,0,0,0,0
26                  BYTE "....",0,0,0,0
27                  BYTE "..",0,0,0,0,0,0
28                  BYTE ".___",0,0,0,0
29                  BYTE "_._",0,0,0,0,0
30                  BYTE "._..",0,0,0,0
31                  BYTE "__",0,0,0,0,0,0
32                  BYTE "_.",0,0,0,0,0,0
33                  BYTE "___",0,0,0,0,0
34                  BYTE ".__.", 0,0,0,0
35                  BYTE "__._", 0,0,0,0
36                  BYTE "._.",0,0,0,0,0
37                  BYTE "...",0,0,0,0,0
38                  BYTE "_",0,0,0,0,0,0,0
39                  BYTE "..._",0,0,0,0,0
40                  BYTE "...._",0,0,0,0
41                  BYTE ".__",0,0,0,0,0
42                  BYTE "_.._",0,0,0,0
43                  BYTE "_.__",0,0,0,0
44                  BYTE  "__..",0,0,0,0
```

0~9 和一些符號初始化，將每個字都設定為 8 個 byte

```
45      Numerals    BYTE "_____",0,0,0
46                  BYTE".____",0,0,0
47                  BYTE"..___",0,0,0
48                  BYTE"...__",0,0,0
49                  BYTE"...._",0,0,0
50                  BYTE".....",0,0,0
51                  BYTE"_....",0,0,0
52                  BYTE"__...",0,0,0
53                  BYTE"___..",0,0,0
54                  BYTE"____.",0,0,0
55      Punctuation BYTE "._._._",0,0
56                  BYTE"__.__",0,0
57                  BYTE"..__..",0,0
58                  BYTE"_.__._",0,0
59                  BYTE".____.",0,0
60                  BYTE"_._._.",0,0
61                  BYTE"___...",0,0
62                  BYTE"._.._.",0,0
63                  BYTE"_...._",0,0
64                  BYTE"_.._.",0,0,0
65                  BYTE"..._.._",0
66
```

主程式，先將一些提示字串印出來，再呼叫 lowertoCap 將校寫的字母轉成大寫的並印出，再呼叫 MorseTran，最後詢問是否再來一次，如果輸入 y 就再跑一次迴圈

```
67      .code
68      main PROC
69      re:
70          mov     edx,OFFSET Welc
71          call    WriteString
72          call    Crlf
73          mov     ecx,BUFMAX
74          mov     edx,OFFSET prompt
75          call    ReadString
76          mov     ecx,eax
77          mov     bufSize,eax
78          mov     eax,edx
79          mov     edx,OFFSET upper
80          call    lowertoCap
81          mov     edx,OFFSET upper
82          call    WriteString
83          call    Crlf
84          call    MorseTran
85          call    Crlf
86          mov     edx,OFFSET Again
87          call    WriteString
88          mov     ecx, BUFMAX
89          mov     edx,OFFSET prompt
90          call    ReadString
91          cmp     byte ptr [prompt],'y'
92          je      re
93          exit
94      main ENDP
95
```

將 eax 的值放到裡然後比較是否在 a~z 之間，如果是就將 assic-32 改成大寫再存到 edx

```
lowertoCap PROC
    ;mov     esi, eax
    ;mov     edi, edx
L1:
    mov     bl, [eax]
    cmp     bl, 'a'
    jb      notLower
    cmp     bl, 'z'
    ja      notLower
    sub     bl, 32
notLower:
    mov     [edx], bl
    inc     eax
    inc     edx
    loop    L1
ret
lowertoCap ENDP
```

先將字串長度存到 ecx，將 eax 指道都是大寫的字串位址，morse 為空字串位址存在 edi，先確認印出的是否為最後，開始比較是否為 A~Z，是就去轉為摩斯密碼，再看是否為 0~9，是就去轉為摩斯密碼，如果不是就會去 ProcessChar 檢查是否是符號

```
MorseTran PROC
    mov     ecx,bufSize
    mov     eax,OFFSET upper
    mov     esi, eax
    mov     edi, OFFSET morse
L2:
    cmp     ecx, 0              ; Check if string ends
    je      EndTranslation
    dec ecx
    mov     al, [esi]
    inc     esi

    cmp     al, 'A'
    jb      CheckDigit
    cmp     al, 'Z'
    jbe     ConvertToMorse
    jmp     ProcessChar

CheckDigit:
    cmp     al, '0'
    jb      CheckSpace
    cmp     al, '9'
    jbe     ConvertToMorseDigit
    jmp     ProcessChar
```

看是否是空格是就跳到 PrintSpace 印出，不是就會去 ProcessChar 檢查是否是符號，157 行開始將大寫字母轉成摩斯密碼，160 行是將現在這個字母和 A 相差多少並乘上 8，因為每個字母存乘 8 個 byte，165 行開始將 0~9 轉成摩斯密碼

```
139     CheckSpace:
140         cmp     al, ' '
141         je      PrintSpace
142         ;jmp     PrintError
143         jmp     ProcessChar
144
145     PrintSpace:
146         mov     al, '/'
147         mov     dl, al
148         call    WriteChar
149         jmp     L2
150
151     PrintError:
152         mov     al, '@'
153         mov     dl, al
154         call    WriteChar
155         jmp     L2
156
157     ConvertToMorse:
158         sub     al, 'A'
159         movzx   ebx, al
160         lea     ebx, [Alphabet + ebx * 8]
161         mov     edx,ebx
162         call    WriteMorse
163         jmp     L2
164
```

```
165     ConvertToMorseDigit:
166         sub     al, '0'
167         movzx   ebx, al
168         lea     ebx, [Numerals + ebx * 8]
169         mov     edx,ebx
170         call    WriteMorse
171         jmp     L2
```

173~288 行都是再確認是否是有效字符，是就轉換成摩斯密碼並 call WriteMorse 印出，否就繼續往下比較，如果都不是就會跳到 PrintError 印出錯誤的@

```
173     ProcessChar:
174         cmp     al, '.'
175         je      Period
176         jmp     next
177     Period:
178         mov     ebx,0
179         lea     ebx, [Punctuation + ebx * 8]
180         mov     edx,ebx
181         call    WriteMorse
182         jmp     L2
183     next:
184         cmp     al, ','
185         je      Comma
186         jmp     next2
187     Comma:
188         mov     ebx,1
189         lea     ebx, [Punctuation + ebx * 8]
190         mov     edx,ebx
191         call    WriteMorse
192         jmp     L2
193     next2:
194         cmp     al, '?'
195         je      Question
196         jmp     next3
197     Question:
198         mov     ebx,2
199         lea     ebx, [Punctuation + ebx * 8]
200         mov     edx,ebx
201         call    WriteMorse
202         jmp     L2
```

```
203     next3:
204         cmp     al, '('
205         je      Parentheses
206         cmp     al, ')'
207         je      Parentheses
208         jmp     next4
209     Parentheses:
210         mov     ebx,3
211         lea     ebx, [Punctuation + ebx * 8]
212         mov     edx,ebx
213         call    WriteMorse
214         jmp     L2
215     next4:
216         cmp     al, 27h
217         je      Apostrophe
218         jmp     next5
219     Apostrophe:
220         mov     ebx,4
221         lea     ebx, [Punctuation + ebx * 8]
222         mov     edx,ebx
223         call    WriteMorse
224         jmp     L2
225     next5:
226         cmp     al, ';'
227         je      Semicolon
228         jmp     next6
229     Semicolon:
230         mov     ebx,5
231         lea     ebx, [Punctuation + ebx * 8]
232         mov     edx,ebx
```

```
233         call    WriteMorse
234         jmp     L2
235     next6:
236         cmp     al, ':'
237         je      Colon
238         jmp     next7
239     Colon:
240         mov     ebx,6
241         lea     ebx, [Punctuation + ebx * 8]
242         mov     edx,ebx
243         call    WriteMorse
244         jmp     L2
245     next7:
246         cmp     al, '"'
247         je      Quotation
248         jmp     next8
249     Quotation:
250         mov     ebx,7
251         lea     ebx, [Punctuation + ebx * 8]
252         mov     edx,ebx
253         call    WriteMorse
254         jmp     L2
255     next8:
256         cmp     al, '-'
257         je      Hyphen
258         jmp     next9
259     Hyphen:
260         mov     ebx,8
261         lea     ebx, [Punctuation + ebx * 8]
262         mov     edx,ebx
```

```
263         call    WriteMorse
264         jmp     L2
265     next9:
266         cmp     al, '/'
267         je      Fraction
268         jmp     next10
269     Fraction:
270         mov     ebx,9
271         lea     ebx, [Punctuation + ebx * 8]
272         mov     edx,ebx
273         call    WriteMorse
274         jmp     L2
275     next10:
276         cmp     al, '$'
277         je      Dollar
278         jmp     PrintError
279     Dollar:
280         mov     ebx,10
281         lea     ebx, [Punctuation + ebx * 8]
282         mov     edx,ebx
283         call    WriteMorse
284         jmp     L2
285
286     EndTranslation:
287     ret
288     MorseTran ENDP
```

將摩斯密碼一個一個印出

```
290     WriteMorse PROC
291         ; EBX contains the address of the Morse code string
292         push        esi
293         mov esi, ebx
294     WriteMorseLoop:
295         mov al, [esi]
296         cmp al, 0
297         je  WriteMorseEnd
298         mov dl, al
299         call WriteChar        ; Use Irvine32's WriteChar to print character
300         inc esi
301         jmp WriteMorseLoop
302     WriteMorseEnd:
303         pop     esi
304         mov     al,' '
305         mov     dl,al
306         call    WriteChar
307         ret
308     WriteMorse ENDP
309
310     END main
```