

Universidad Autónoma de Baja California  
Facultad de Ciencias Químicas e Ingeniería



## **ORGANIZACIÓN DE LAS COMPUTADORAS Y LENGUAJE ENSAMBLADOR**

### **Practica 8**

**Ejercicios básicos de entrada/salida en el lenguaje  
ensamblador procesador 8086**

**Docente:** Sanchez Herrera Mauricio Alonso

**Alumno:** Gómez Cárdenas Emmanuel Alberto

**Matricula:** 1261509

## Contenido

TEORIA.....	4
Interrupciones.....	4
Capturar desde teclado .....	4
Mostrar a pantalla .....	4
DESARROLLO.....	5
Parte 1 (Usando PCLIB06.LIB).....	5
Hola mundo.....	5
Capturar y mostrar un dígito.....	5
Capturar un dígito hexadecimal y mostrarlo como decimal.....	6
Escribir un programa que muestre una caja de asteriscos (*) de tamaño 5x5. ....	6
Escribir un programa que genere el siguiente patrón.....	6
Capturar dos letras en mayúscula y mostrarlas en orden alfabético.....	7
Capturar una letra mayúscula y otra minúscula indicando cuál es cual. ....	8
Escribir un programa que en base al nombre del animal este escriba la onomatopeya. ....	9
Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa: .....	10
Parte 2 (Sin usar PCLIB06.LIB).....	11
Hola mundo.....	11
Capturar y mostrar un dígito.....	11
Capturar un dígito hexadecimal y mostrarlo como decimal.....	12
Escribir un programa que muestre una caja de asteriscos (*) de tamaño 5x5. ....	12
Escribir un programa que genere el siguiente patrón.....	12
Capturar dos letras en mayúscula y mostrarlas en orden alfabético.....	13
Capturar una letra mayúscula y otra minúscula indicando cuál es cual. ....	14
Escribir un programa que en base al nombre del animal este escriba la onomatopeya. ....	15
Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa: .....	16
CONCLUSIONES.....	17
REFERENCIAS.....	17
ANEXOS .....	18

Códigos de la parte 2 (Sin usar PCLIB06).....	18
Hola mundo. ....	18
Capturar y mostrar un dígito. ....	19
Capturar un dígito hexadecimal y mostrarlo como decimal. ....	21
Escribir un programa que muestre una caja de asteriscos (*) de tamaño 5x5.....	23
Escribir un programa que genere el siguiente patrón. ....	24
Capturar dos letras en mayúscula y mostrarlas en orden alfabético. ....	26
Capturar una letra mayúscula y otra minúscula indicando cuál es cual.....	29
Escribir un programa que en base al nombre del animal este escriba la onomatopeya. .....	32
Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa: .....	38

## TEORIA

### Interrupciones

Una interrupción se da cuando se interrumpe un programa que está en ejecución. Esto con el fin de resolver un proceso que es considerado más importante. Al resolver el proceso, el procesador vuelve al programa que había sido interrumpido y continua justo donde se suspendió. Estas pueden ser llamadas por los periféricos o por software.

Para Interactuar con el usuario, se utilizan las interrupciones como una interfaz humano-maquina. Dependiendo de la interrupción utilizada es el tipo de interfaz que se va a generar.

### Capturar desde teclado

Para leer desde el teclado existen las interrupciones:

- 10h/Servicio 08h: Lee un char y su atributo en la posición del cursor.
- 16h/Servicio 00h: Obtiene una pulsación del teclado y la elimina del buffer.
- 16h/Servicio 01h: Revisa por una pulsación del teclado, no la remueve del buffer.
- 21h/Servicio 01h: Lee un char desde la entrada estándar.
- 21h/Servicio 06h: Entrada o salida directa a consola.
- 21h/Servicio 07h: Entrada de carácter sin eco a AL
- 21h/Servicio 0Ah: Entrada de una String a DS:DX, el primer byte es el tamaño del buffer y el segundo es el número de chars leídos.
- 21h/Servicio 0Ch: Limpia el buffer de teclado y lee la entrada estándar

### Mostrar a pantalla

- 10h/Servicio 00h: Selecciona el modo de video (2 opciones modo texto y 1 modo gráfico)
- 10h/Servicio 01h: Selecciona la forma del cursor de modo texto.
- 10h/Servicio 09h: Escribe un char y atributos en la posición del cursor.
- 10h/Servicio 0Ah: Escribe solo el char en la posición del cursor.
- 10h/Servicio 0Ch: Cambia el color de solo un pixel.
- 10h/Servicio 0Eh: Salida de teletipo.
- 10h/Servicio 13h: Escribe una String.
- 10h/Servicio 0Ah: Escribe solo el char en la posición del cursor.
- 21h/Servicio 02h: Escribe un char a la salida estándar.
- 21h/Servicio 06h: Entrada o salida directa a consola.
- 21h/Servicio 09h: Salida de una String en DS:DX. Esta debe terminar con '\$'.

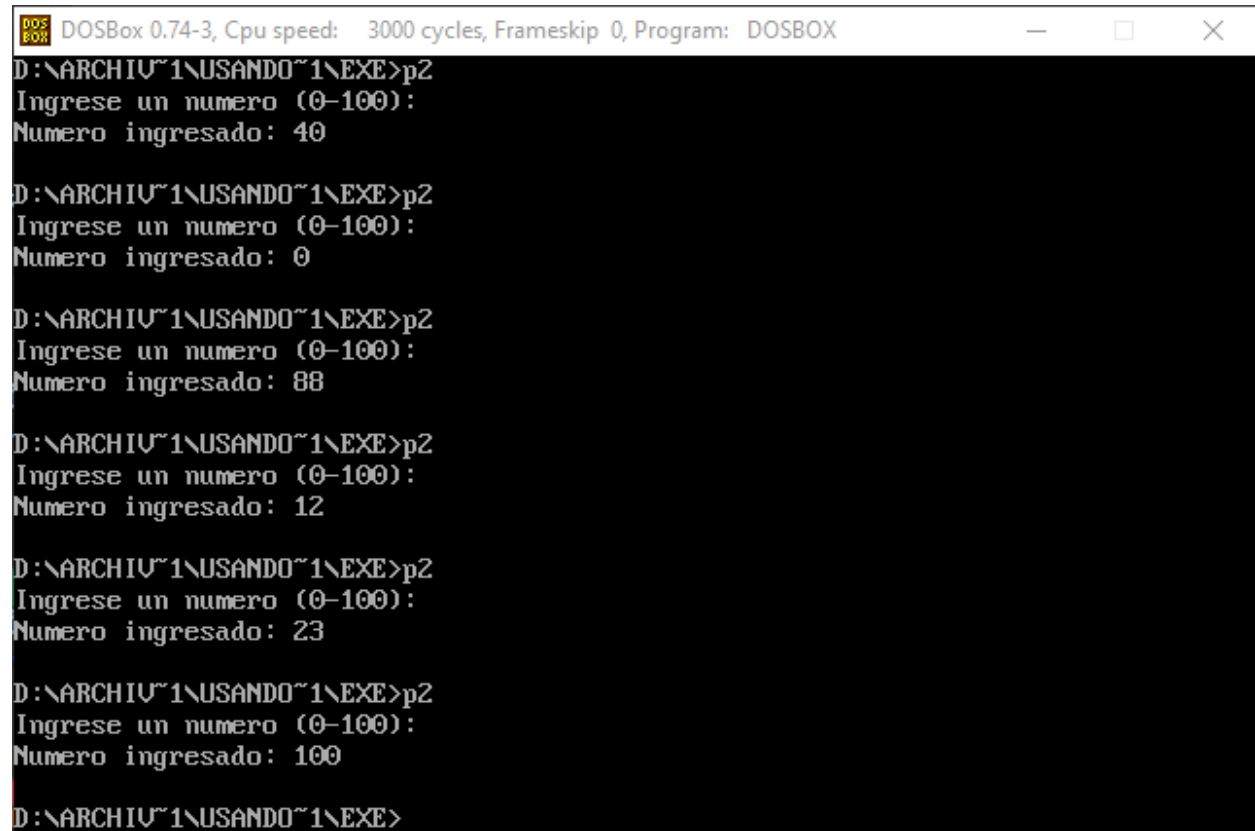
## DESARROLLO

### Parte 1 (Usando PCLIB06.LIB)

Hola mundo

```
D:\ARCHIVO~1\USANDO~1\EXE>p1  
Hola Mundo
```

Capturar y mostrar un dígito.



The screenshot shows a DOSBox window titled "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The command prompt inside shows the following sequence of commands and outputs:

```
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 40  
  
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 0  
  
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 88  
  
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 12  
  
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 23  
  
D:\ARCHIVO~1\USANDO~1\EXE>p2  
Ingrese un numero (0-100):  
Numero ingresado: 100  
  
D:\ARCHIVO~1\USANDO~1\EXE>
```

Capturar un dígito hexadecimal y mostrarlo como decimal.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): F
Dígito ingresado en decimal: 15

D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): a
Dígito ingresado no válido...

D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): 2
Dígito ingresado en decimal: 2

D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): D
Dígito ingresado en decimal: 13

D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): B
Dígito ingresado en decimal: 11

D:\ARCHIVO~1\USANDO~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): s
Dígito ingresado no válido...

D:\ARCHIVO~1\USANDO~1\EXE>S
  
```

Escribir un programa que muestre una caja de asteriscos (\*) de tamaño 5x5.

```

D:\ARCHIVO~1\USANDO~1\EXE>p4

*****
*****
*****
*****
*****
  
```

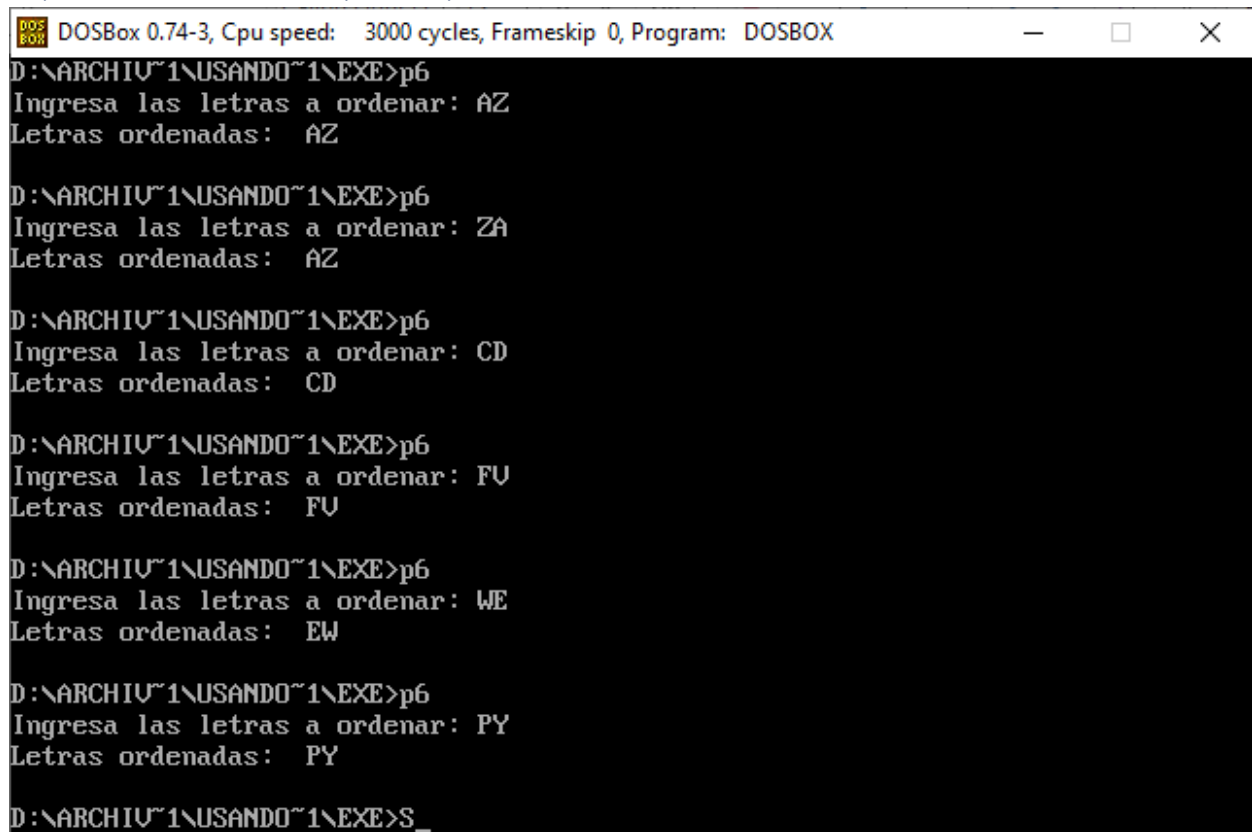
Escribir un programa que genere el siguiente patrón.

```

*
**
***
****
*****
*****
****
***
**
*

D:\ARCHIVO~1\USANDO~1\EXE>p5
  
```

Capturar dos letras en mayúscula y mostrarlas en orden alfabético.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: AZ
Letras ordenadas:  AZ

D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: ZA
Letras ordenadas:  AZ

D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: CD
Letras ordenadas:  CD

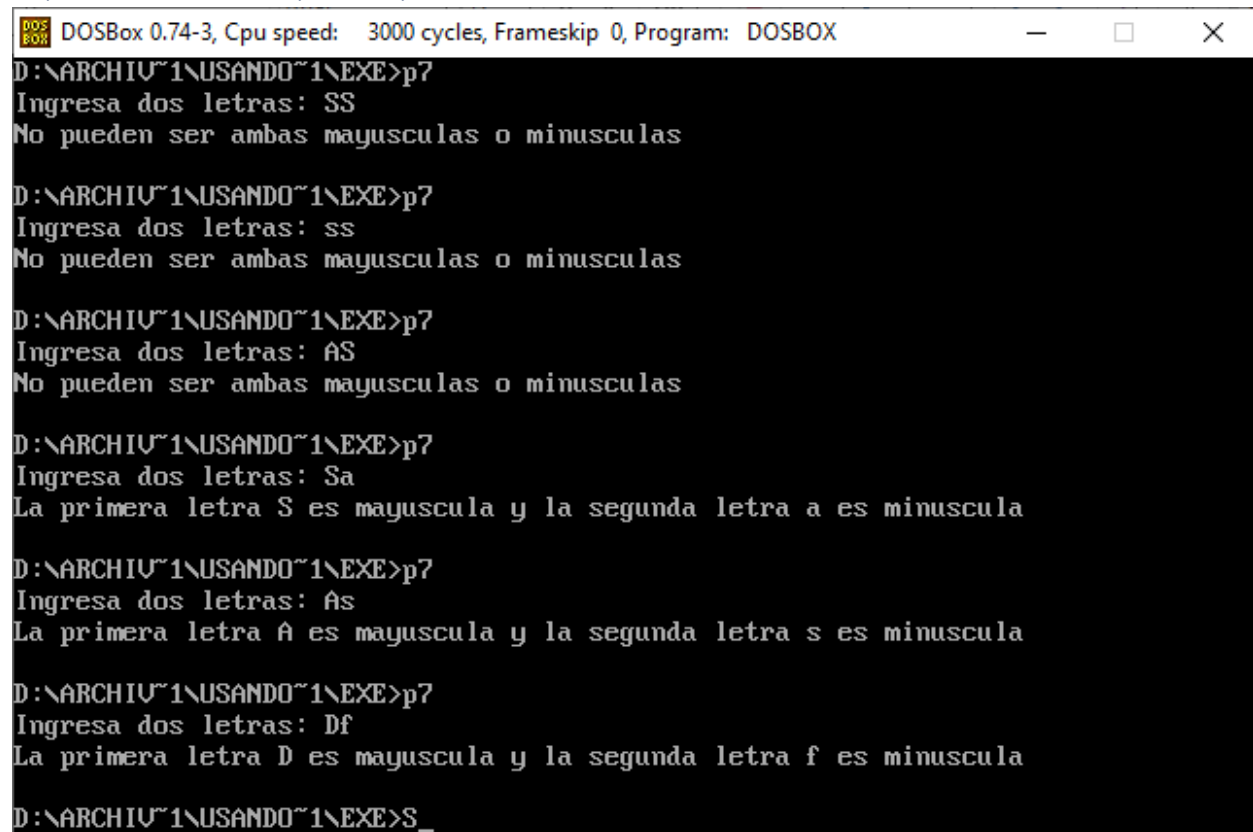
D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: FU
Letras ordenadas:  FU

D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: WE
Letras ordenadas:  EW

D:\ARCHIVO~1\USANDO~1\EXE>p6
Ingresa las letras a ordenar: PY
Letras ordenadas:  PY

D:\ARCHIVO~1\USANDO~1\EXE>S_
```

Capturar una letra mayúscula y otra minúscula indicando cuál es cual.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: SS
No pueden ser ambas mayusculas o minusculas

D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: ss
No pueden ser ambas mayusculas o minusculas

D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: AS
No pueden ser ambas mayusculas o minusculas

D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: Sa
La primera letra S es mayuscula y la segunda letra a es minuscula

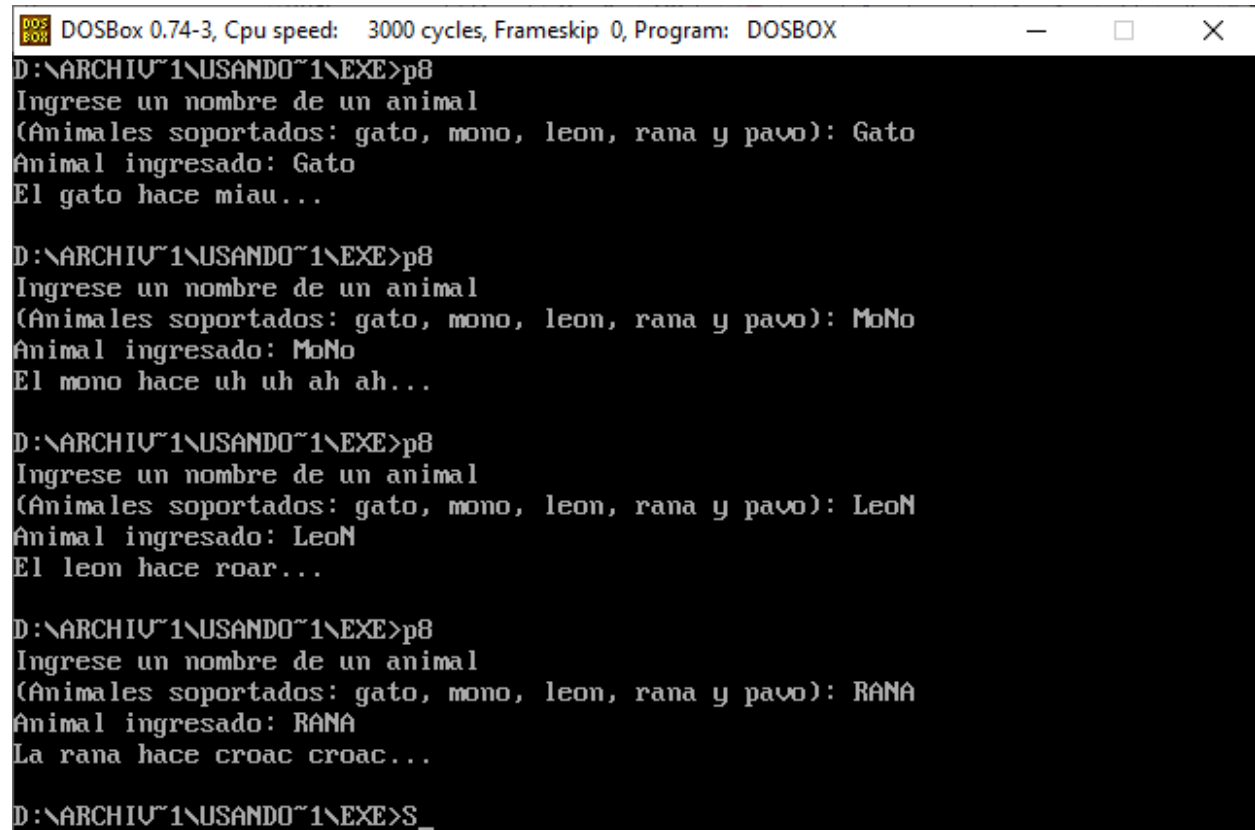
D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: As
La primera letra A es mayuscula y la segunda letra s es minuscula

D:\ARCHIVO~1\USANDO~1\EXE>p7
Ingresa dos letras: Df
La primera letra D es mayuscula y la segunda letra f es minuscula

D:\ARCHIVO~1\USANDO~1\EXE>S_
```



Escribir un programa que en base al nombre del animal este escriba la onomatopeya.



```
DOSBOX 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO~1\USANDO~1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): Gato
Animal ingresado: Gato
El gato hace miau...

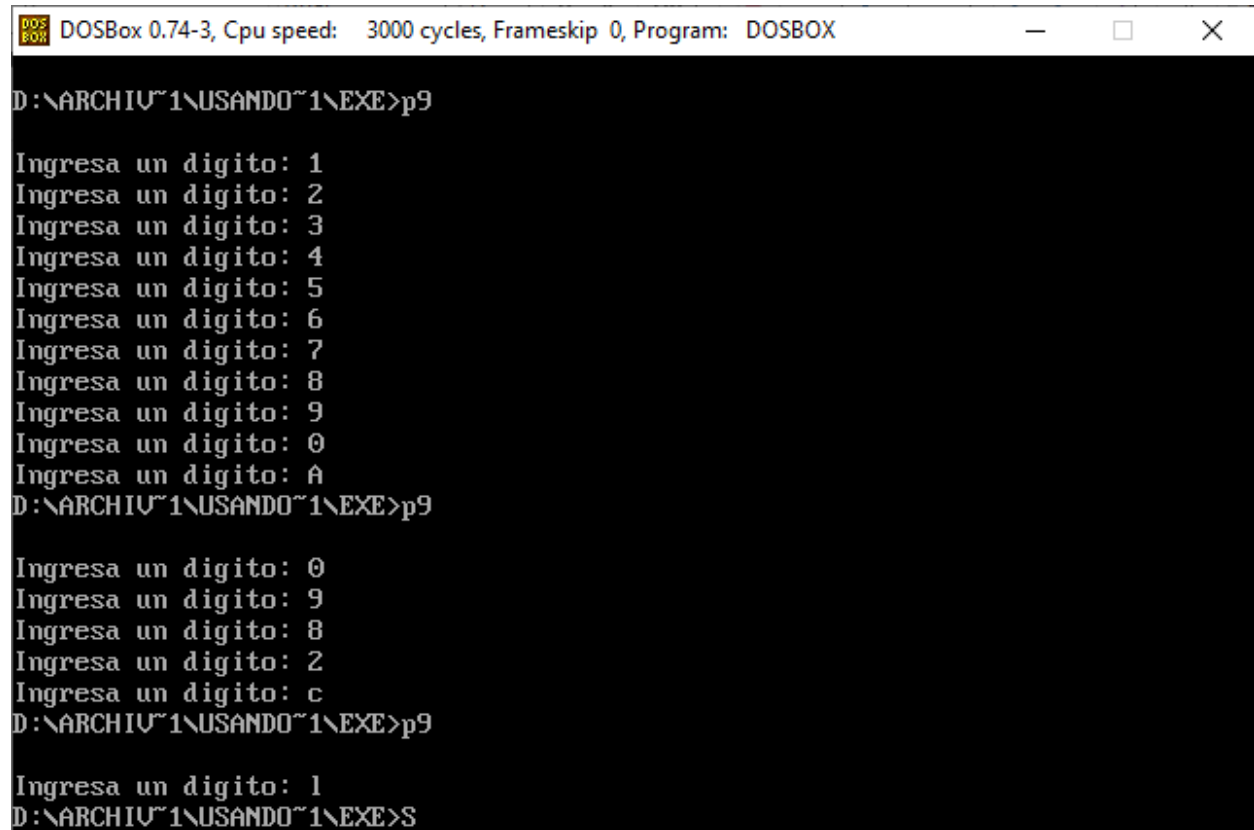
D:\ARCHIVO~1\USANDO~1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): MoNo
Animal ingresado: MoNo
El mono hace uh uh ah ah...

D:\ARCHIVO~1\USANDO~1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): LeoN
Animal ingresado: LeoN
El leon hace roar...

D:\ARCHIVO~1\USANDO~1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): RANA
Animal ingresado: RANA
La rana hace croac croac...

D:\ARCHIVO~1\USANDO~1\EXE>S_
```

Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa:



```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

D:\ARCHIVO~1\USANDO~1\EXE>p9

Ingresa un dígito: 1
Ingresa un dígito: 2
Ingresa un dígito: 3
Ingresa un dígito: 4
Ingresa un dígito: 5
Ingresa un dígito: 6
Ingresa un dígito: 7
Ingresa un dígito: 8
Ingresa un dígito: 9
Ingresa un dígito: 0
Ingresa un dígito: A
D:\ARCHIVO~1\USANDO~1\EXE>p9

Ingresa un dígito: 0
Ingresa un dígito: 9
Ingresa un dígito: 8
Ingresa un dígito: 2
Ingresa un dígito: c
D:\ARCHIVO~1\USANDO~1\EXE>p9

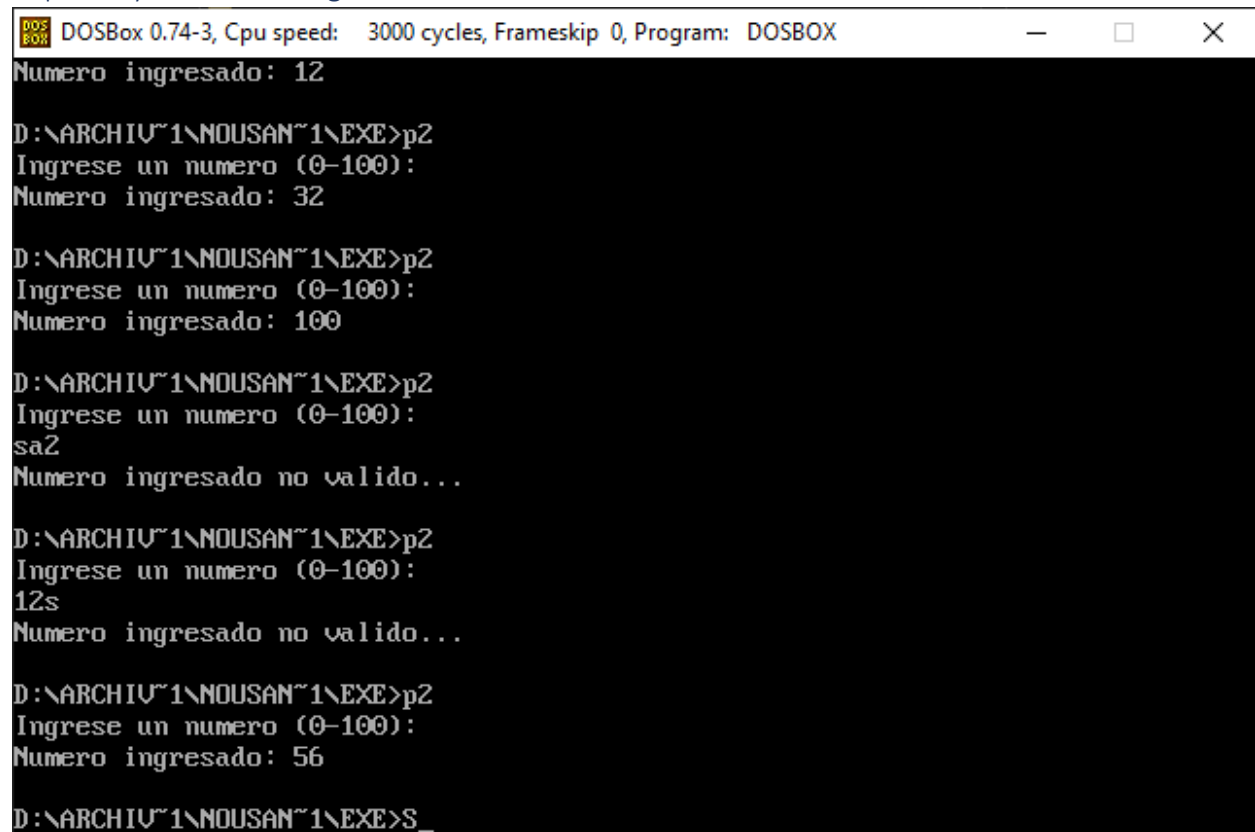
Ingresa un dígito: l
D:\ARCHIVO~1\USANDO~1\EXE>S
```

## Parte 2 (Sin usar PCLIB06.LIB)

Hola mundo

```
D:\ARCHIVO~1\NOUSAN~1\EXE>p1
Hola Mundo, impreso con los servicios BIOS
Hola Mundo, impreso con los servicios DOS
```

Capturar y mostrar un dígito.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Numero ingresado: 12

D:\ARCHIVO~1\NOUSAN~1\EXE>p2
Ingrese un numero (0-100):
Numero ingresado: 32

D:\ARCHIVO~1\NOUSAN~1\EXE>p2
Ingrese un numero (0-100):
Numero ingresado: 100

D:\ARCHIVO~1\NOUSAN~1\EXE>p2
Ingrese un numero (0-100):
sa2
Numero ingresado no valido...

D:\ARCHIVO~1\NOUSAN~1\EXE>p2
Ingrese un numero (0-100):
12s
Numero ingresado no valido...

D:\ARCHIVO~1\NOUSAN~1\EXE>p2
Ingrese un numero (0-100):
Numero ingresado: 56

D:\ARCHIVO~1\NOUSAN~1\EXE>S_
```

Capturar un dígito hexadecimal y mostrarlo como decimal.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): A
Dígito ingresado en decimal: 10

D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): C
Dígito ingresado en decimal: 12

D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): 3
Dígito ingresado en decimal: 3

D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): 6
Dígito ingresado en decimal: 6

D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): f
Dígito ingresado no valido...

D:\ARCHIVO~1\NOUSAN~1\EXE>p3
Ingrese un dígito en hexadecimal (0-F): F
Dígito ingresado en decimal: 15

D:\ARCHIVO~1\NOUSAN~1\EXE>S_
  
```

Escribir un programa que muestre una caja de asteriscos (\*) de tamaño 5x5.

```

D:\ARCHIVO~1\NOUSAN~1\EXE>p4

*****
*****
*****
*****
*****
  
```

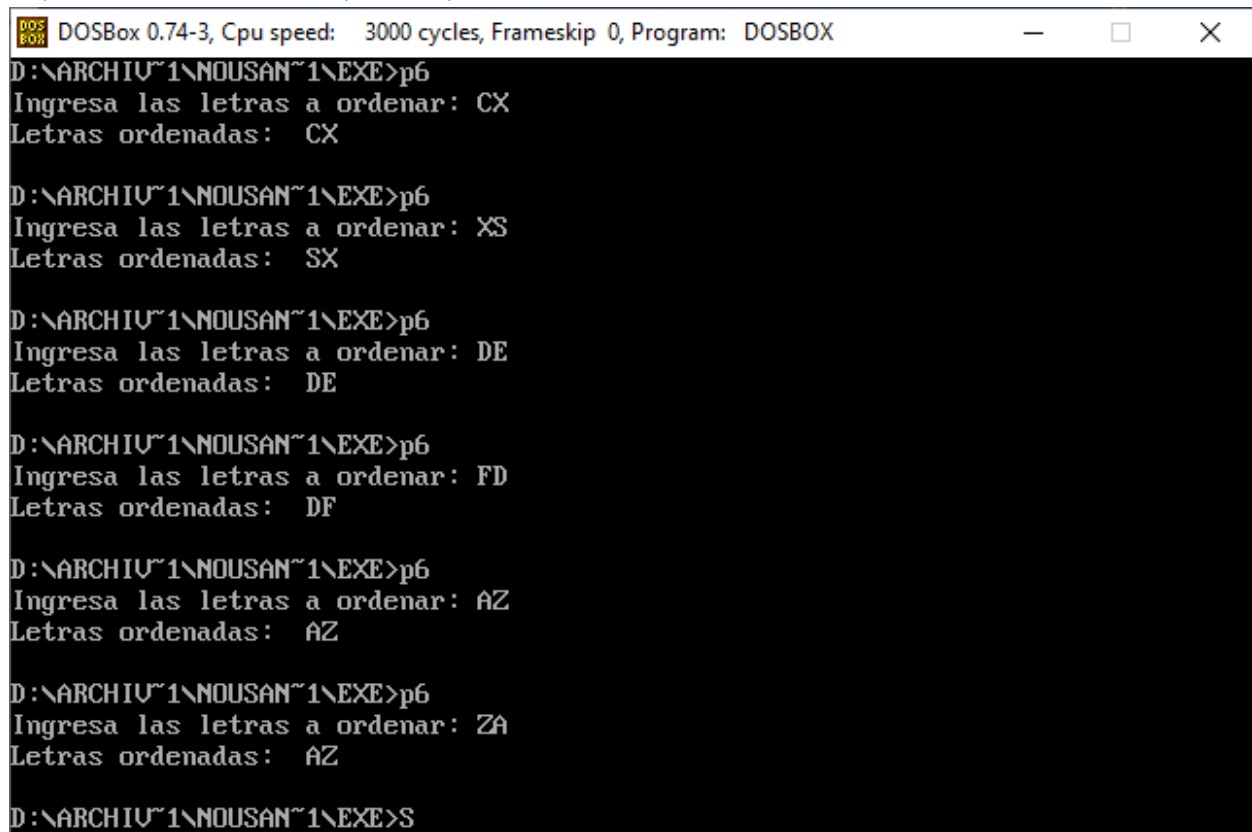
Escribir un programa que genere el siguiente patrón.

```

*
**
***
****
*****
****
***
**
*

D:\ARCHIVO~1\NOUSAN~1\EXE>p5
  
```

Capturar dos letras en mayúscula y mostrarlas en orden alfabético.



```
DOSBOX 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: CX
Letras ordenadas: CX

D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: XS
Letras ordenadas: SX

D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: DE
Letras ordenadas: DE

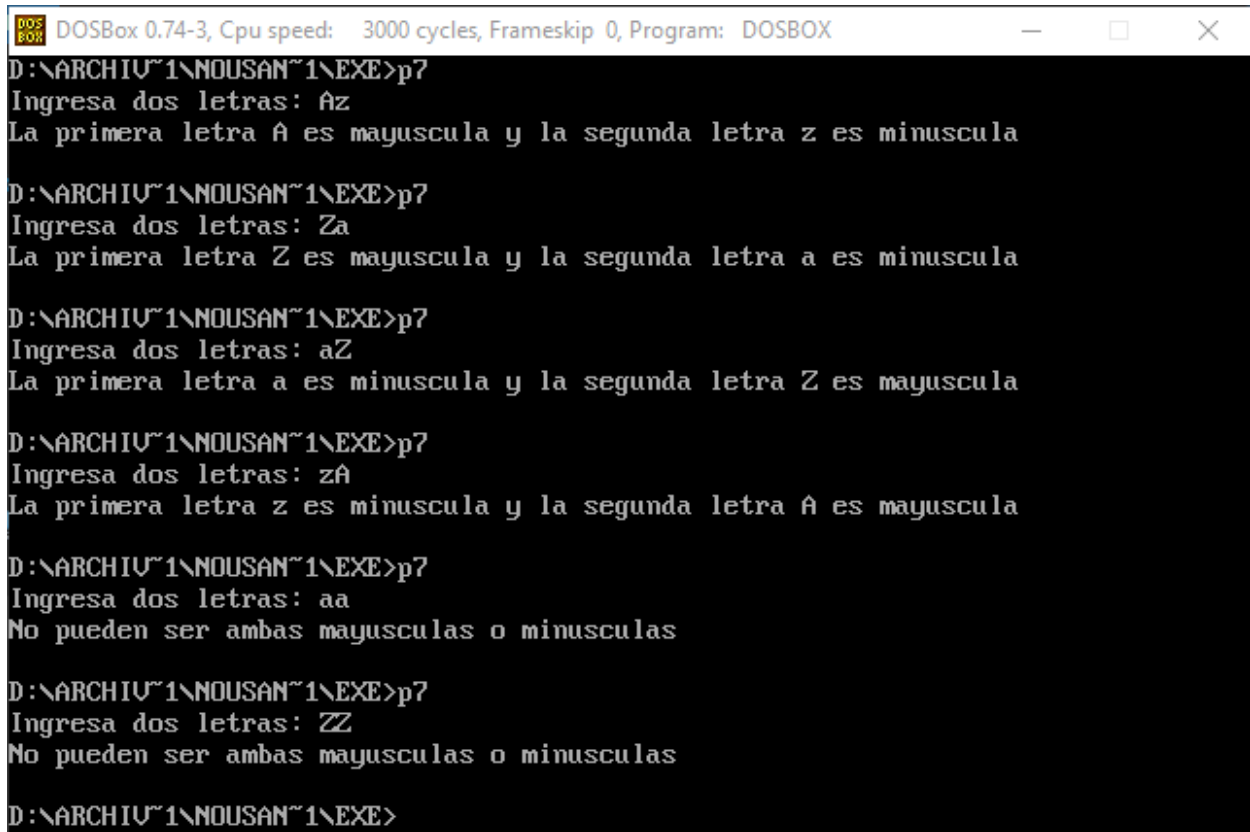
D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: FD
Letras ordenadas: DF

D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: AZ
Letras ordenadas: AZ

D:\ARCHIVO\1\NOUSAN\1\EXE>p6
Ingresa las letras a ordenar: ZA
Letras ordenadas: AZ

D:\ARCHIVO\1\NOUSAN\1\EXE>S
```

Capturar una letra mayúscula y otra minúscula indicando cuál es cual.



```
DOSBOX 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: Az
La primera letra A es mayuscula y la segunda letra z es minuscula

D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: Za
La primera letra Z es mayuscula y la segunda letra a es minuscula

D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: aZ
La primera letra a es minuscula y la segunda letra Z es mayuscula

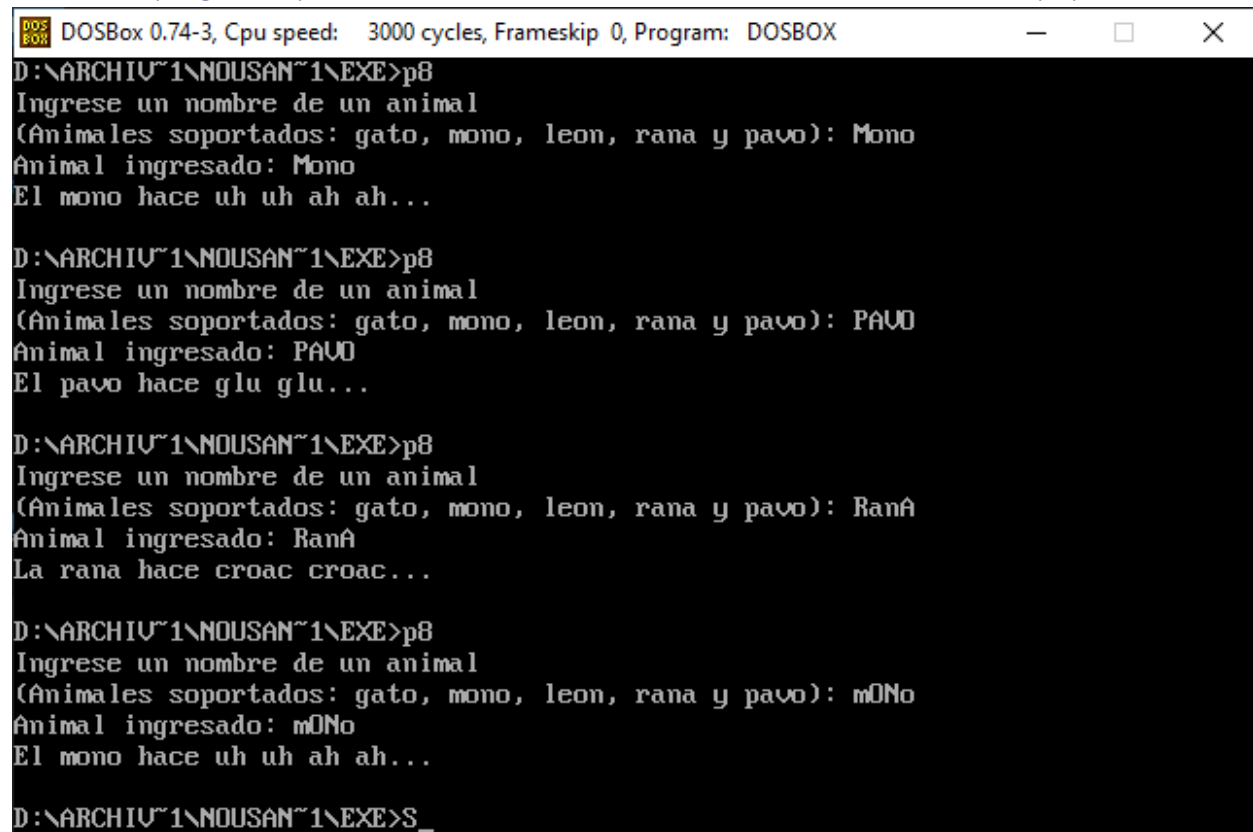
D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: zA
La primera letra z es minuscula y la segunda letra A es mayuscula

D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: aa
No pueden ser ambas mayusculas o minusculas

D:\ARCHIVO\1\NOUSAN\1\EXE>p7
Ingresa dos letras: ZZ
No pueden ser ambas mayusculas o minusculas

D:\ARCHIVO\1\NOUSAN\1\EXE>
```

Escribir un programa que en base al nombre del animal este escriba la onomatopeya.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
D:\ARCHIVO\1\NOUSAN\1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): Mono
Animal ingresado: Mono
El mono hace uh uh ah ah...

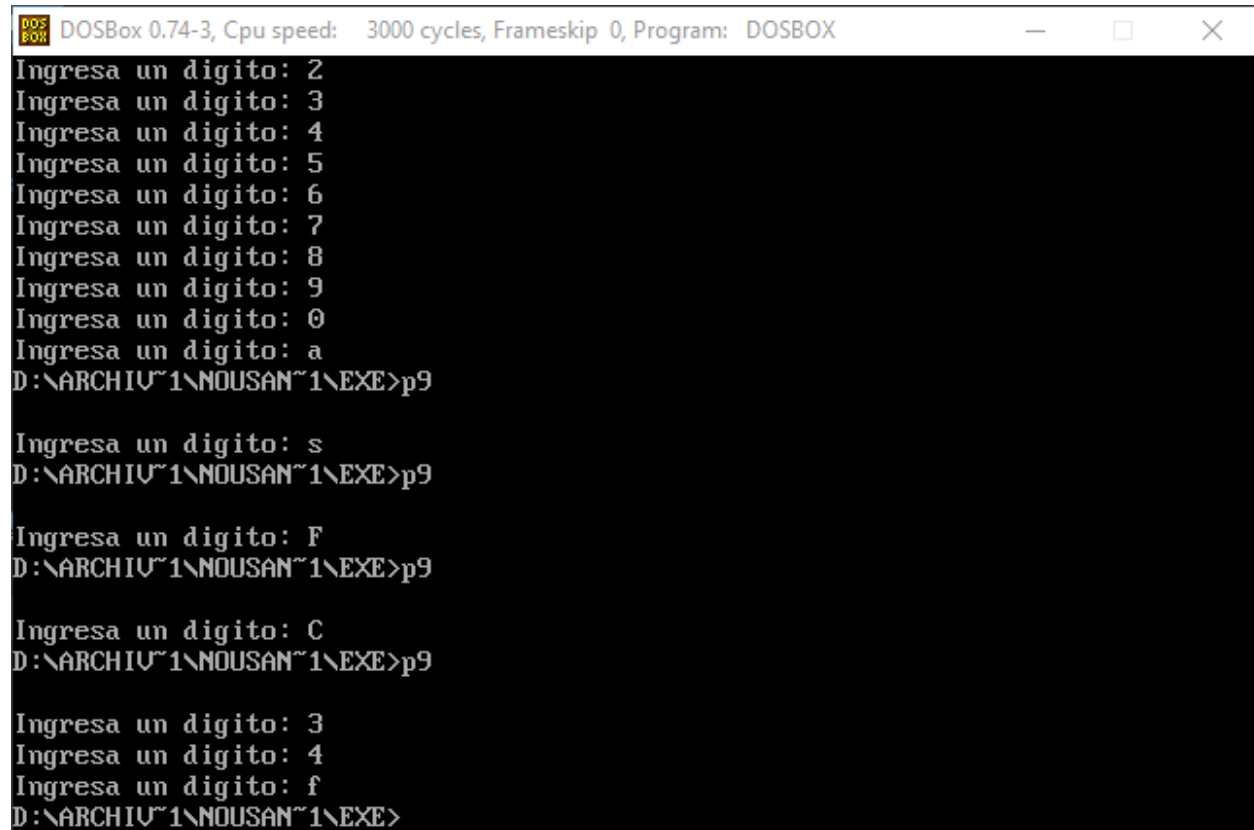
D:\ARCHIVO\1\NOUSAN\1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): PAVO
Animal ingresado: PAVO
El pavo hace glu glu...

D:\ARCHIVO\1\NOUSAN\1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): Rana
Animal ingresado: Rana
La rana hace croac croac...

D:\ARCHIVO\1\NOUSAN\1\EXE>p8
Ingrese un nombre de un animal
(Animales soportados: gato, mono, leon, rana y pavo): mONO
Animal ingresado: mONO
El mono hace uh uh ah ah...

D:\ARCHIVO\1\NOUSAN\1\EXE>S_
```

Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Ingresa un digito: 2
Ingresa un digito: 3
Ingresa un digito: 4
Ingresa un digito: 5
Ingresa un digito: 6
Ingresa un digito: 7
Ingresa un digito: 8
Ingresa un digito: 9
Ingresa un digito: 0
Ingresa un digito: a
D:\ARCHIVO~1\NOUSAN~1\EXE>p9

Ingresa un digito: s
D:\ARCHIVO~1\NOUSAN~1\EXE>p9

Ingresa un digito: F
D:\ARCHIVO~1\NOUSAN~1\EXE>p9

Ingresa un digito: C
D:\ARCHIVO~1\NOUSAN~1\EXE>p9

Ingresa un digito: 3
Ingresa un digito: 4
Ingresa un digito: f
D:\ARCHIVO~1\NOUSAN~1\EXE>
```



## CONCLUSIONES

Es muy interesante como las interrupciones sirven de interfaz humano-máquina para poder compartir información, que existan bastantes interrupciones es algo bueno, ya que, si un servicio de “x” interrupción no se ajusta a tus necesidades, puedes buscar un servicio de otra interrupción que haga básicamente lo mismo, pero de otra forma.

## REFERENCIAS

*2 Assembly Language Programming*. Cs.unm.edu. (2020). Retrieved from <https://www.cs.unm.edu/~maccabe/classes/341/labman/node2.html>.

## ANEXOS

Códigos de la parte 2 (Sin usar PCLIB06)

Hola mundo.

```

MODEL small
.STACK 100h
LOCALS
.DATA
    endASK    db  'Presione cualquier tecla para salir...',0
    helloBIOS db  'Hola Mundo, impreso con los servicios BIOS ',10,13,0
    helloDOS  db  'Hola Mundo, impreso con los servicios DOS ',10,13,0
.CODE
Principal    PROC
    mov ax,@data    ;Inicializar DS al la direccion
    mov ds,ax       ; del segmento de datos (.DATA)

    mov di, offset helloBIOS

@@printBIOS:    ;Impresion utilizando el servicio 0Eh de la
                ; interrupcion 10h de BIOS

    mov al,[di]
    cmp al,0h    ;Cuando se encuentre el caracter de terminacion
    je @@continue ;Termina de imprimir
    mov ah,0Eh    ;Servicio 0Eh
    int 10h       ;Interrupcion 10h
    inc di        ;Aumenta el indice de datos
    jmp @@printBIOS ;Regresa al comienzo

@@continue:    mov di, offset helloDOS
@@printDOS:    ;Impresion utilizando el servicio 02h de la
                ; interrupcion 21h de DOS

    mov dl,[di]
    cmp dl,0h    ;Cuando se encuentre el caracter de termiacion
    je @@endPrinting ;Termina de imprimir
    mov ah,02h    ;Servicio 02h
    int 21h       ;Interrupcion 21h
    inc di        ;Aumenta el indice de datos
    jmp @@printDOS ;Regresa al comienzo

@@endPrinting:
    mov ah,04ch    ; fin de programa
    mov al,0
    int 21h
    ENDP
END

```

## Capturar y mostrar un dígito.

```

MODEL small
.STACK 100h

LOCALS

.DATA
buffer db 4,?,4 dup(0)
ask db 'Ingrese un numero (0-100): ',10,13,0
result db 'Numero ingresado: ',0
xceptn db 10,13,'Numero ingresado no valido...',0
newln db 10,13,0

.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data ;Inicializar DS al la direccion
    mov ds,ax ; del segmento de datos (.DATA)

    mov di,offset ask
    call putStr

    mov dx, offset buffer
    mov ah, 0Ah
    int 21h
    mov bl,buffer[1] ;Despues de utilizar la interrupcion 21 es necesario
    mov buffer[bx+2], 0;agregar el caracter de terminacion manualmente
    ;El tamaño del string se encuentra en la segunda posicion

    mov al, buffer[2]
    cmp buffer[2],'9' ;Si el primer char es mayor que 39Ah (ascii de 9)
    jg @@exception ;manda error
    cmp buffer[2],'0' ;Si el primer char es menor que 30h (ascii de 0)
    jl @@exception ;manda error
    cmp buffer[3],0h ;Si el segundo char es 0
    je @@continue ;salta a continuar sin hacer mas pruebas
    cmp buffer[3],'9' ;hace las mismas comprobaciones para el 2do char
    jg @@exception
    cmp buffer[3],'0'
    jl @@exception
    cmp buffer[4],0h ;Si el tercer char es 0
    je @@continue ;salta a continuar sin hacer mas pruebas
    cmp buffer[4],'9' ;hace las mismas comprobaciones para el 3er char
    jg @@exception
    cmp buffer[4],'0'
    jl @@exception
@@continue:
    mov di,offset result
    call putStr

    mov di,offset buffer+2 ;Debido a que la interrupción 21/0Ah utiliza
    ;los dos primeros bytes para guardar información, es necesario

```

```
        call putStr      ;desplazar el resultado 2 posiciones
        jmp @@endProgram
@@exception:
        mov di,offset xceptn
        call putStr
@@endProgram:
        mov di,offset newln
        call putStr
        mov ah,04ch      ; fin de programa
        mov al,0          ;
        int 21h          ;
        ENDP

putStr  PROC      ;Funcion para imprimir un string
        push ax          ;Guarda registros a modificar
        push di
@@putStr:
        mov al,[di]      ;Bbtiene el valor de la direccion di y la guarda en al
        cmp al,0h        ;Si es caracter de terminacion
        je @@endputStr   ; Dejar de imprimir
        mov ah,0Eh       ;Selecciona el servicio 0Eh
        int 10h          ;Llama la interrupcion 10h
        inc di           ;decrementa di
        jmp @@putStr
@@endputStr:
        pop di           ;Recupera registros modificados
        pop ax
        ret
        ENDP
END
```

Capturar un dígito hexadecimal y mostrarlo como decimal.

```

MODEL small
.STACK 100h

;----- Insert INCLUDE "filename" directives here
;----- Insert EQU and = equates here

LOCALS

.DATA
ask db 'Ingrese un dígito en hexadecimal (0-F): ',0
result db 'Dígito ingresado en decimal: ',0
xceptn db 'Dígito ingresado no valido...',0
endASK db 'Presione cualquier tecla para salir...',0

.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data    ;Inicializar DS al la direccion
    mov ds,ax       ; del segmento de datos (.DATA)

    mov di,offset ask
    call putStr
    call getChar
    call putNewline

    cmp al,'0'       ;comprueba que sea mayor o igual a 30h (0 en ascii)
    jl @@exception
    cmp al,'F'       ;comprueba que sea menor o igual a 46h (F en ascii)
    jg @@exception
    cmp al,3Ah       ;Si es menor que 3Ah se salta las pruebas porque el
                    ;numero esta entre (0-9)
    jl @@endCheck

    cmp al,'A'       ;Si es menor que 'A'
    jl @@exception ;Manda excepcion

    sub al,11h       ;Se le resta 11h para indicar su valor (F-11=5)
    push ax          ;Se guarda el valor de AX
    mov di,offset result
    call putStr      ;Mensaje de respuesta
    mov al,'1'       ;Se imprime el '1'
    call putChar
    pop ax           ;Se recupera el valor de ax
    call putChar     ;Y se imprime
    jmp @@endProgram

@@endCheck:
    ;Si el valor es un numero, se imprime directamente
    mov di,offset result
    call putStr
    call putChar

```

```

        jmp @@endProgram
@@exception:
    mov di,offset xceptn
    call putStr
@@endProgram:
    mov ah,04ch      ; fin de programa
    mov al,0         ;
    int 21h          ;

        ENDP

putStr  PROC        ;Funcion para imprimir un string
        push ax      ;Guarda registros a modificar
        push di
@@putStr:
    mov al,[di]      ;Obtiene el valor de la direccion di y la guarda en al
    cmp al,0h        ;Si es caracter de terminacion
    je @@endputStr   ; Dejar de imprimir
    mov ah,0Eh       ;Selecciona el servicio 0Eh
    int 10h          ;Llama la interrupcion 10h
    inc di            ;decrementa di
    jmp @@putStr
@@endputStr:
    pop di            ;Recupera registros modificados
    pop ax
    ret
        ENDP

putNewline PROC      ;Funcion para imprimir un salto de linea
    push ax
    mov al, 0Ah      ;Salto de linea
    call putChar
    mov al, 0Dh      ;Retorno de carro
    call putChar
    pop ax
    ret
        ENDP

putChar PROC         ;Funcion para imprimir un char almacenado en al
    push ax          ;Guarda el valor del registro a modificar
    mov ah,0Eh       ;Selecciona el servicio 0Eh
    int 10h          ;Llama la interrupcion 10h
    pop ax           ;Recupera registro
    ret
        ENDP

getChar PROC         ;Funcion para leer un char y almacenarlo en al
    mov ah,1h        ;Selecciona el servicio 01h
    int 21h          ;Llama la interrupcion 21h
    ret
        ENDP
END

```

Escribir un programa que muestre una caja de asteriscos (\*) de tamaño 5x5.

```

MODEL small
.STACK 100h
LOCALS

.DATA

.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data ;Inicializar DS al la direccion
    mov ds,ax    ; del segmento de datos (.DATA)
    mov cx,0h
    mov al,2Ah

@@for1:
    cmp ch,5h    ;Cuando CH sea 5
    je @@endProgram ;Sale del programa
    inc ch
    xor cl,cl    ;Resetea el valor de cl
    call putNewline

@@for2:
    call putChar
    cmp cl,4h    ;Imprime el char 5 veces
    je @@for1
    inc cl
    jmp @@for2

@@endProgram:
    call putNewline
    mov ah,04ch  ; fin de programa
    mov al,0
    int 21h

    ENDP

putNewline PROC ;Funcion para imprimir un salto de linea
    push ax
    mov al, 0Ah ;Salto de linea
    call putChar
    mov al, 0Dh ;Retorno de carro
    call putChar
    pop ax
    ret
    ENDP

putChar PROC ;Funcion para imprimir un char almacenado en al
    push ax ;Guarda el valor del registro a modificar
    mov ah,0Eh ;Selecciona el servicio 0Eh
    int 10h ;Llama la interrupcion 10h
    pop ax ;Recupera registro
    ret
    ENDP
END

```

Escribir un programa que genere el siguiente patrón.

```
*
**
***
****
*****
*****
****
***
**
*
```

```
MODEL small
.STACK 100h
LOCALS
.DATA
.CODE
Principal PROC
    mov ax,@data ;Inicializar DS al la direccion
    mov ds,ax ; del segmento de datos (.DATA)
    mov cx,0h
    mov al,2Ah
    mov bh,5h
    mov bl,1h

@@for1:
    cmp ch,bh
    je @@changeSign;Cambia el signo de bl
    add ch,bl
    mov cl,1h ;Reinicia el valor de cl en 1
    call putNewline

@@for2:
    cmp ch,0h
    je @@endProgram
    call putChar
    cmp cl,ch ;Imprime el valor CH veces
    je @@for1
    inc cl
    jmp @@for2

@@changeSign:
    cmp ch,0h ;Si CH es 0
    je @@endProgram;Sale del programa
    sub bl,2 ;Se hace negativo
    mov bh,0h ;Se hace 0
    jmp @@for1

@@endProgram:
    mov ah,04ch ; fin de programa
    mov al,0 ;
    int 21h ;
    ENDP

putNewline PROC ;Funcion para imprimir un salto de linea
    push ax
```



```
        mov al, 0Ah    ;Salto de linea
        call putChar
        mov al, 0Dh    ;Retorno de carro
        call putChar
        pop ax
        ret
    ENDP
putChar PROC          ;Funcion para imprimir un char almacenado en al
    push ax           ;Guarda el valor del registro a modificar
    mov ah,0Eh        ;Selecciona el servicio 0Eh
    int 10h           ;Llama la interrupcion 10h
    pop ax            ;Recupera registro
    ret
ENDP
END
```

Capturar dos letras en mayúscula y mostrarlas en orden alfabético.

```

MODEL small
.STACK 100h

;----- Insert INCLUDE "filename" directives here
;----- Insert EQU and = equates here

LOCALS

.DATA
message db 'Ingresa las letras a ordenar: ',0
exceptn db 'Ambas letras deben ser mayusculas...',0
sorted db 'Letras ordenadas: ',0,10,13
charA db 0
charB db 0

.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data ;Inicializar DS al la direccion
    mov ds,ax ; del segmento de datos (.DATA)

    mov di,offset message
    call putStr

    call getChar ;Lee un valor y lo guarda en charA
    mov charA,al
    call getChar ;Lee un valor y lo guarda en charB
    mov charB,al
    call isAplha ;Compara que charB sea mayuscula
    cmp bl,01h
    jne @@exception
    mov al,charA ;Compara que charA sea mayuscula
    call isAplha
    cmp bl,01h
    jne @@exception

    call putNewline
    mov di,offset sorted
    call putStr

    cmp al,charB
    jl @@bGreater
    mov al,charB ;Si charA es mas grande

```

```

        call putChar    ;Imprime primero B y
        mov  al,charA   ;luego A
        call putChar
        jmp  @@endProgram

@@bGreater: call putChar    ;Si B es mas grande, imprime A y luego B
        mov  al,charB
        call putChar
        jmp  @@endProgram

@@exception:call putNewline
        mov  di,offset xceptn
        call putStr

@@endProgram:
        mov  ah,04ch      ; fin de programa
        mov  al,0
        int  21h

        ENDP

putStr  PROC      ;Funcion para imprimir un string
        push ax          ;Guarda registros a modificar
        push di
@@putStr:
        mov  al,[di]     ;Obtiene el valor de la direccion di y la guarda en al
        cmp  al,0h       ;Si es caracter de terminacion
        je   @@endputStr ; Dejar de imprimir
        mov  ah,0Eh      ;Selecciona el servicio 0Eh
        int  10h         ;Llama la interrupcion 10h
        inc  di          ;decrementa di
        jmp  @@putStr

@@endputStr:
        pop  di          ;Recupera registros modificados
        pop  ax
        ret

        ENDP

putNewline PROC    ;Funcion para imprimir un salto de linea
        push ax
        mov  al, 0Ah     ;Salto de linea
        call putChar
        mov  al, 0Dh     ;Retorno de carro
        call putChar
        pop  ax
        ret

        ENDP

```

```

putChar  PROC      ;Funcion para imprimir un char almacenado en al
            push ax      ;Guarda el valor del registro a modificar
            mov ah,0Eh    ;Selecciona el servicio 0Eh
            int 10h       ;Llama la interrupcion 10h
            pop ax       ;Recupera registro
            ret
        ENDP

getChar  PROC      ;Funcion para leer un cahar y almacenarlo en al
            mov ah,1h     ;Selecciona el servicio 01h
            int 21h       ;Llama la interrupcion 21h
            ret
        ENDP

;Comprueba que al se encuentre dentro del alfabeto
isAplha  PROC      ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, si es minuscula
            00h
            push ax
            ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')
            cmp al,'A'
            jnl @@exception
            cmp al,'Z'
            jnl @@isUpper
            ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')
            cmp al,'a'
            jnl @@exception
            cmp al,'z'
            jnl @@isLower
            ;Si no se cumple ningun caso anterior, no esta en el alfabeto
            jmp @@exception
@@exception:
            mov bl,2h
            jmp @@endCheck
@@isUpper:
            mov bl,1h
            jmp @@endCheck
@@isLower:
            mov bl,0h
@@endCheck:
            pop ax      ;Recupera registro
            ret
        ENDP
END

```

Capturar una letra mayúscula y otra minúscula indicando cuál es cual.

```

MODEL small
.STACK 100h
LOCALS
.DATA
message db 'Ingresa dos letras: ',0
xceptn db 'Ambas letras deben estar en el alfabeto (A-z)...',0
message1 db 'La primera letra ',0
message2 db 'y la segunda letra ',0
upper db ' es mayuscula ',0
lower db ' es minuscula ',0
charA db 0
charB db 0
.CODE
Principal PROC
    mov ax,@data ;Inicializar DS al la direccion
    mov ds,ax ; del segmento de datos (.DATA)

    mov di,offset message
    call putStr

    call getChar ;Lee un valor para charA
    mov charA,al
    call getChar ;Lee un valor para charB
    mov charB,al
    call putNewline
    mov al,charA
    call isAlpha ;Revisa si esta en el alfabeto
    cmp bl,02h ;Si no esta en el alfabeto
    je @@exception ;Manda una excepcion
    mov bh,bl ;Copia bl a bh

    mov al,charB
    call isAlpha ;Revisa si esta en el alfabeto

    cmp bl,02h ;Si no esta en el alfabeto
    je @@exception ;Manda una excepcion
    cmp bl,bh ;Si ambas letras son mayuculas o minusculas
    je @@exception2
    mov cl,0h ;Reinicia el contador
    mov di,offset message1
    call putStr
    mov al,charA ;Imprime charA
    call putchar
    cmp bh,01h ;Si es mayuscula
    je @@printupper ;Imprime mensaje mayuscula
    jmp @@printlower ;Imprime mensaje minuscula
@@second:
    ;Ahora imprime charB
    mov di,offset message2
    call putStr
    mov al,charB

```

```

        call putchar
        cmp bl,01h
        je @@printupper
        jg @@exception
        jmp @@printlower

@@printupper: ;Imprime diciendo que es mayuscula
        mov di,offset upper
        call putStr
        inc cl
        cmp cl,1h
        je @@second
        jmp @@endProgram

@@printlower: ;Imprime diciendo que es minuscula
        mov di,offset lower
        call putStr
        inc cl
        cmp cl,1h
        je @@second
        jmp @@endProgram

@@exception:
        call putNewline
        mov di,offset xceptn
        call putStr
        jmp @@endProgram

@@exception2:
        call putNewline
        mov di,offset xceptn2
        call putStr

@@endProgram:
        mov ah,04ch      ; fin de programa
        mov al,0          ;
        int 21h           ;

        ENDP

putStr PROC      ;Funcion para imprimir un string
        push ax      ;Guarda registros a modificar
        push di
@@putStr:
        mov al,[di]   ;Obtiene el valor de la direccion di y la guarda en al
        cmp al,0h     ;Si es caracter de terminacion
        je @@endputStr ; Dejar de imprimir
        mov ah,0Eh    ;Selecciona el servicio 0Eh
        int 10h       ;Llama la interrupcion 10h
        inc di        ;decrementa di
        jmp @@putStr
@@endputStr:
        pop di        ;Recupera registros modificados

```

```

        pop ax
        ret
    ENDP

putNewline PROC    ;Funcion  para imprimir un salto de linea
    push ax
    mov al, 0Ah    ;Salto de linea
    call putChar
    mov al, 0Dh    ;Retorno de carro
    call putChar
    pop ax
    ret
    ENDP

putChar  PROC      ;Funcion para imprimir un char almacenado en al
    push ax        ;Guarda el valor del registro a modificar
    mov ah,0Eh     ;Selecciona el servicio 0Eh
    int 10h        ;Llama la interrupcion 10h
    pop ax         ;Recupera registro
    ret
    ENDP

getChar  PROC      ;Funcion para leer un cahar y almacenarlo en al
    mov ah,1h      ;Selecciona el servicio 01h
    int 21h        ;Llama la interrupcion 21h
    ret
    ENDP

;Comprueba que al se encuentre dentro del alfabeto
isAlpha PROC ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, 00h
    push ax
    ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')
    cmp al,'A'
    jl @@exception
    cmp al,'Z'
    jle @@isUpper
    ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')
    cmp al,'a'
    jl @@exception
    cmp al,'z'
    jle @@isLower
    ;Si no se cumple ningun caso anterior, no esta en el alfabeto
    jmp @@exception
@@exception: mov bl,2h
              jmp @@endCheck
@@isUpper:   mov bl,1h
              jmp @@endCheck
@@isLower:   mov bl,0h
@@endCheck:  pop ax    ;Recupera registro
              ret
    ENDP
END

```

Escribir un programa que en base al nombre del animal este escriba la onomatopeya.

```

MODEL small
.STACK 100h

;----- Insert INCLUDE "filename" directives here
;----- Insert EQU and = equates here

LOCALS

.DATA
animal db 5,?,5 dup(0)
ask db 'Ingresa un nombre de un animal', 10,13,'(Animales soportados: gato, mono, le
on, rana y pavo): ',0
answer db 'Animal ingresado: ',0
exceptn db 'Animal ingresado no soportado...',0
cat db 'El gato hace miau...',0
monkey db 'El mono hace uh uh ah ah...',0
lion db 'El leon hace roar...',0
frog db 'La rana hace croac croac...',0
turkey db 'El pavo hace glu glu...',0
gato db 'gato',0
mono db 'mono',0
leon db 'leon',0
rana db 'rana',0
pavo db 'pavo',0

.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data    ;Inicializar DS al la direccion
    mov ds,ax       ; del segmento de datos (.DATA)

    mov di,offset ask    ;Pide que se ingrese un animal
    call putStr
    mov dx,offset animal ;Pone la etiqueta en la que se guardara lo que se lea
    call getStr          ;Lee del teclado

    call putNewline
    mov di,offset answer ;Muestra el mensaje de que se ha capturado algo
    call putStr

    mov di,offset animal+2 ;Muestra el animal capturado
    call putStr
    call compareString     ;Compara el string capturado di, con los animales soportados
                           ;el resultado se guarda en bl
    cmp bl,2h              ;Si el resultado es 2h, no se encontro ningun animal
                           ;soportado

```



```

        je @@exception

        ;bl indica el animal encontrado y se imprime su mensaje
        cmp bl,3h
        je @@gato

        cmp bl,4h
        je @@mono

        cmp bl,5h
        je @@leon

        cmp bl,6h
        je @@rana

        cmp bl,7h
        je @@pavo
        ;si no se encuentra un animal, imprime una excepcion
        ;(Esta linea no deberia suceder)
        ;pero se pone solo por si se llega a dar el caso
        jmp @@exception

@@gato:  mov di, offset cat
        jmp @@print
@@mono:  mov di, offset monkey
        jmp @@print
@@leon:  mov di, offset lion
        jmp @@print
@@rana:  mov di, offset frog
        jmp @@print
@@pavo:  mov di, offset turkey
        jmp @@print
@@exception:
        mov di,offset xceptn
@@print: call putNewline    ;Imprime el salto de linea y el mensaje en di
        call putStr
@@endProgram:
        call putNewline
                mov ah,04ch    ; fin de programa
                mov al,0
                int 21h
ENDP

getStr   PROC
        mov ah, 0Ah          ;Se selecciona el servicio 0Ah
        int 21h              ;Se llama a la interrupcion 21h
        mov bl,animal[1]     ;Despues de utilizar la interrupcion 21h es necesario
        mov animal[bx+2], 0   ;agregar el caracter de terminacion manualmente
        ENDP                 ;El tamaño del string se encuentra en la segunda posicion

compareString PROC
        push di               ;Guarda el valor del reg di

```

```

    call toLower          ;Convierte todas las mayusculas encontradas a minusculas
    cmp bl,2h             ;Si se encuentra una exception
                        ;(al menos un char no esta dentro del alfabeto)
    je @@endCmp           ; Termina la comparacion
    call compareAnimal    ;En cualquier otro caso, se compara el str uno a uno con los an
imales
@@endCmp:
    pop di               ;Se recupera el registro usado
    ret
    ENDP

compareAnimal PROC
    push ax              ;Se guardan los registros a utilizar
    push cx
    push di
    push dx
    mov cx,0h           ;Se resetea CX
    mov al,[di]          ;Se obtiene el primer char de la cadena en DI
    cmp al,'g'           ;Si es 'g'
    je @@gato            ;Se continua la comparacion con gato
    cmp al,'m'           ;Si es 'm'
    je @@mono            ;Se continua la comparacion con mono
    cmp al,'l'           ;Si es 'l'
    je @@leon            ;Se continua la comparacion con leon
    cmp al,'r'           ;Si es 'r'
    je @@rana            ;Se continua la comparacion con rana
    cmp al,'p'           ;Si es 'p'
    je @@pavo            ;Se continua la comparacion con pavo
    jmp @@exception      ;En cualquier otro caso, manda una excepcion
@@gato:
    inc di              ;Incrementa di y cx
    inc cx
    mov al,[di]          ;Se obtiene el valor de di
    cmp al,0h           ;Si es un caracter de terminacion
    je @@gatoCheck       ;Revisa el resultado
    mov bx,cx            ;Se copia el valor de cx en bx
    cmp al,gato[bx]      ;Se otiene el char de gato, desplazado por bx (si bx=1, se obtiene
una 'a')
    je @@gato            ;Si son iguales, sigue comparando
    jmp @@exception      ;En cualquier otro caso, manda una excepcion
@@gatoCheck:
    cmp cx,04h           ;Debido a que los animales son de 4 letras, cx debe terminar en 4
    jne @@exception
    mov bl, 03h ;03h indica que el gato ha sido encontrado
    jmp @@endCompare
@@mono:
    inc di
    inc cx
    mov al,[di]
    cmp al,0h
    je @@monoCheck
    mov bx,cx

```

```
    cmp al,mono[bx]
    je @@mono
    jne @@exception
@@monoCheck:
    cmp cx,04h
    jne @@exception
    mov bl, 04h ;04h indica que el mono ha sido encontrado
    jmp @@endCompare
@@leon:
    inc di
    inc cx
    mov al,[di]
    cmp al,0h
    je @@leonCheck
    mov bx,cx
    cmp al,leon[bx]
    je @@leon
    jne @@exception
@@leonCheck:
    cmp cx,04h
    jne @@exception
    mov bl, 05h ;05h indica que el leon ha sido encontrado
    jmp @@endCompare
@@rana:
    inc di
    inc cx
    mov al,[di]
    cmp al,0h
    je @@ranaCheck
    mov bx,cx
    cmp al,rana[bx]
    je @@rana
    jne @@exception
@@ranaCheck:
    cmp cx,04h
    jne @@exception
    mov bl, 06h ;06h indica que la rana ha sido encontrada
    jmp @@endCompare
@@pavo:
    inc di
    inc cx
    mov al,[di]
    cmp al,0h
    je @@pavoCheck
    mov bx,cx
    cmp al,pavo[bx]
    je @@pavo
    jne @@exception
@@pavoCheck:
    cmp cx,04h
    jne @@exception
    mov bl, 07h ;07h indica que el pavo ha sido encontrado
```

```

        jmp @@endCompare
@@exception:
        mov bl,02h
        jmp @@endCompare
@@endCompare:
        pop dx
        pop di
        pop cx
        pop ax
        ret
        ENDP

;Comprueba que al se encuentre dentro del alfabeto
isAlpha PROC ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h
              ;si es minuscula 00h
        push ax
        ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')
        cmp al,'A'
        jl @@exception
        cmp al,'Z'
        jl @@isUpper
        ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')
        cmp al,'a'
        jl @@exception
        cmp al,'z'
        jl @@isLower
        ;Si no se cumple ningun caso anterior, no esta en el alfabeto
        jmp @@exception
@@exception:
        mov bl,2h
        jmp @@endCheck
@@isUpper:
        mov bl,1h
        jmp @@endCheck
@@isLower:
        mov bl,0h
@@endCheck:
        pop ax ;Recupera registro
        ret
        ENDP

toLower PROC ;Convierte todas las mayusculas encontradas a minusculas
        push ax
        push di
@@convStr:
        mov al,[di]
        cmp al,0h ;Si el caracter es el de terminacion
        je @@endConvStr ;Termina conversion
        call isAlpha ;Se revisa si esta en el abecedario y si es mayuscula o minuscula
        cmp bl,2h ;Si no se encuentra en el abecedario
        je @@endConvStr ;Se deja de convertir
        cmp bl,01h ;Si es mayuscula

```

```

    je @@upper2Lower    ;Convierte a minuscula
    inc di              ;Incrementa di
    jmp @@convStr       ;Sigue convirtiendo
@@upper2Lower:
    add al,' '          ;Se le suma ' ' que equivale a la diferencia entre 'A' y 'a' (20h)
    mov [di],al         ;Se guarda el char, ahora en minuscula
    inc di              ;Incrementa di
    jmp @@convStr       ;Sigue convirtiendo
@@endConvStr:
    pop di              ;Recupera registros utilizados
    pop ax
    ret
    ENDP

putStr    PROC          ;Funcion para imprimir un string
    push ax             ;Guarda registros a modificar
    push di
@@putStr:
    mov al,[di]         ;Obtiene el valor de la direccion di y la guarda en al
    cmp al,0h           ;Si es caracter de terminacion
    je @@endputStr      ; Dejar de imprimir
    mov ah,0Eh          ;Selecciona el servicio 0Eh
    int 10h             ;Llama la interrupcion 10h
    inc di              ;decrementa di
    jmp @@putStr
@@endputStr:
    pop di              ;Recupera registros modificados
    pop ax
    ret
    ENDP

putNewline PROC         ;Funcion para imprimir un salto de linea
    push ax
    mov al, 0Ah         ;Salto de linea
    call putChar
    mov al, 0Dh         ;Retorno de carro
    call putChar
    pop ax
    ret
    ENDP

putChar   PROC          ;Funcion para imprimir un char almacenado en al
    push ax             ;Guarda el valor del registro a modificar
    mov ah,0Eh          ;Selecciona el servicio 0Eh
    int 10h             ;Llama la interrupcion 10h
    pop ax              ;Recupera registro
    ret
    ENDP
END

```

Escribir un programa que te permita capturar números de un dígito hasta detectar alguna letra, minúscula o mayúscula, caso que parará el programa:

```

MODEL small
.STACK 100h

;----- Insert INCLUDE "filename" directives here
;----- Insert EQU and = equates here

LOCALS

.DATA
message db 'Ingresa un dígito: ',0
.CODE
;----- Insert program, subroutine call, etc., here

Principal PROC
    mov ax,@data    ;Inicializar DS al la direccion
    mov ds,ax       ; del segmento de datos (.DATA)

@@loop:
    call putNewline
    mov di,offset message
    call putStr
    call getChar
    call isDigit
    cmp bl,0h
    je @@endProgram
    jmp @@loop

@@endProgram:
    mov ah,04ch      ; fin de programa
    mov al,0         ;
    int 21h          ;

    ENDP

putStr PROC          ;Funcion para imprimir un string
    push ax          ;Guarda registros a modificar
    push di

@@putStr:
    mov al,[di]       ;Obtiene el valor de la direccion di y la guarda en al
    cmp al,0h         ;Si es caracter de terminacion
    je @@endputStr    ; Dejar de imprimir
    mov ah,0Eh        ;Selecciona el servicio 0Eh
    int 10h           ;Llama la interrupcion 10h
    inc di            ;decrementa di

```

```

        jmp @@putStr
@@endputStr:
        pop di          ;Recupera registros modificados
        pop ax
        ret
    ENDP

putNewline PROC    ;Funcion para imprimir un salto de linea
    push ax
    mov al, 0Ah     ;Salto de linea
    call putChar
    mov al, 0Dh     ;Retorno de carro
    call putChar
    pop ax
    ret
    ENDP

putChar PROC       ;Funcion para imprimir un char almacenado en al
    push ax        ;Guarda el valor del registro a modificar
    mov ah,0Eh     ;Selecciona el servicio 0Eh
    int 10h        ;Llama la interrupcion 10h
    pop ax         ;Recupera registro
    ret
    ENDP

getChar PROC       ;Funcion para leer un char y almacenarlo en al
    mov ah,1h      ;Selecciona el servicio 01h
    int 21h        ;Llama la interrupcion 21h
    ret
    ENDP

isDigit PROC ; < -- Indica a TASM el inicio del un procedimiento
    cmp al,'9'
    jg @@exception
    cmp al,'0'
    jl @@exception
    mov bl,1h
    jmp @@endCheck
@@exception:
    mov bl,0h
@@endCheck:
    ret
    ENDP; < -- Indica a TASM el fin del procedimiento

END

```

06/MAYO/2020

Gómez Cárdenas Emmanuel Alberto