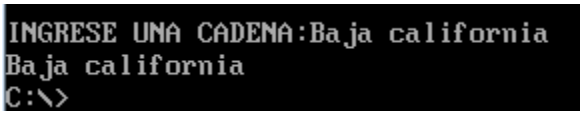
Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

Alumno: Eduardo Marcelo Gutiérrez Soto
Profesora: Evangelina Lara Camacho
Tarea 2

Procedimiento GETS

Ejecución:



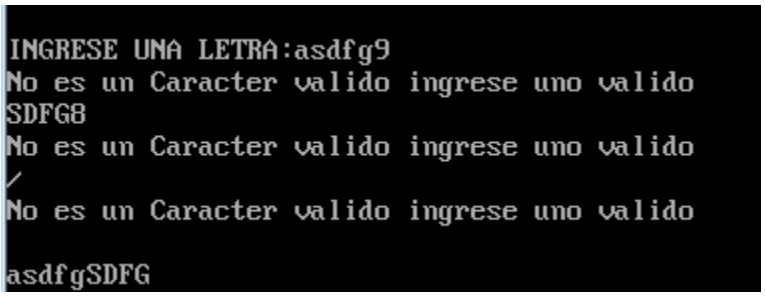
Código:

```
GETS PROC
push ax
    push cx
    push bx
    PUSH DX
    mov cl,0
@@captura:mov ah,01h;capturamos el primer caracter
     int 21h
@@borrar:cmp al,8       ;verificamos si es el backspace
     jne @@enter    ;si no es igual saltamos a verificar si se preciono un enter
     cmp cl,0       ;comparamos si cl es 0,si lo es entonces no hay elemento que borrar
     mov dl," "     ;funcion para eliminar el eco de la pantalla
     mov ah,02h          ;
     int 21h
     je @@captura   ;si cl es 0 entonces brincamos a la captura otra vez
     mov byte ptr[bx],0 ;en caso que si se preciono el y cl no sea 0 backspace borramos el elemento
     dec bx             ;decrementamos bx para apuntar a la direccion bx-1 un elemento atras
     inc cl             ;incrementamos cl por el eco agregado para eliminar el elemento en pantalla
     mov dl,8           ;pasamos a al el backspace para retroceder un elemento hacia atras
     mov ah,02h
     int 21h
     jmp @@captura      ;volvemos a capturar un caracter
@@enter: cmp al,13          ;si el caracter ingresado es enter salimos de la capturacion de la cadena
     je @@fin
     mov [bx],al        ;si no es igual pasamos el elemento al al contenido de [bx]
     inc bx             ;incrementamos bx para apuntar a la siguiente direccion
     dec cl             ;decremento cl para cuando se utilize un backspace no se atore en el mismo elemento
     jmp @@captura
@@fin:   mov byte ptr[bx],0 ;ya que capturo toda su cadena dejamos el ultimo elemento con un 0

    POP DX
    pop bx
    pop cx
    pop ax
    ret
```

Procedimiento getsAlpha

Ejecución:



Código:

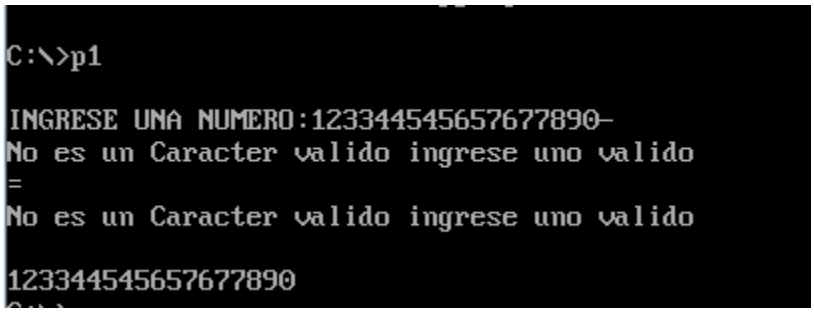
```
;procedimiento que recibe caracteres del abecedario
;entre a hasta la z, y de A hasta la Z
        getsALPHA proc
        push ax
            push cx
            push bx
            PUSH DX
            mov cl,0
    @@captura:mov ah,01h    ;capturamos el primer caracter
              int 21h
             ;probamos que solo sean caracteres permitidos.
             test al,13 ;enter
             jz @@borrar
             test al,8  ;backspace
             jz @@borrar
             test al,32 ;espacio
             jz @@borrar
             ;ahora hacemos las pruebas con letras

             cmp al,65  ;a mayus
             jb @@no_valido     ;limitamos a numeros menores a 65
             cmp al,90  ;z mayus
             jbe @@borrar       ;limitamos a numeros menores a 90

             cmp al,122 ;z minus
             ja @@no_valido     ;limitamos a numeros mayores a 122
             cmp al,97  ;a minus
             jb @@no_valido     ;limitamos a numeros menores a 97
             jmp @@borrar       ;cumplidas las condiciones anteriores capturamos otro letra
    @@no_valido:
                mov dx,offset MENS_E
                call puts
                jmp @@captura
```

Procedimiento getsNum

Ejecución:



Código:

```
;procedimiento que captura solo numeros
;entre el 0 y el 9

           getsNUM proc
            push ax
            push cx
            push bx
            PUSH DX

            mov cl,0
    @@captura:mov ah,01h;capturamos el primer caracter
              int 21h
             ;probamos que solo sean caracteres permitidos.
             test al,13 ;enter
             jz @@borrar
             test al,8  ;backspace
             jz @@borrar
             test al,32 ;espacio
             jz @@borrar
             ;ahora hacemos las pruebas con numeros

             cmp al,57      ;hacemos una comparacion con el numero 9
             ja @@no_valido
             cmp al,48      ;hacemos una comparacion con el numero 0
             jb @@no_valido
             jmp @@borrar   ;si las condiciones se cumplen seguimos captur
    @@no_valido:
             mov dx,offset MENS_E
             call puts
             jmp @@captura
```

Procedimiento toUpperCase

Ejecución:

```
C:\>tlink p1,,,pclib06
Turbo Link  Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\>p1

PROCEDIMIENO QUE CONVIERTE UNA CADENA A MAYUSCULA

INGRESE UNA CADENA:hola profesora evangelina
HOLA PROFESORA EVANGELINA
C:\>
```

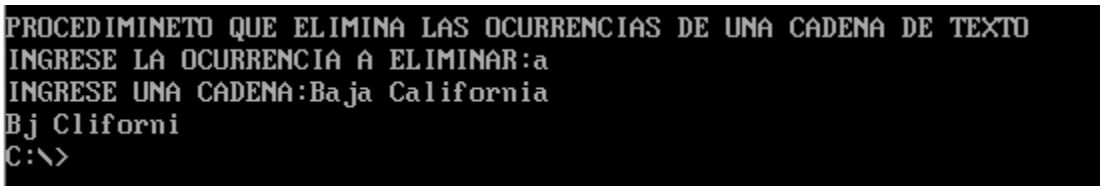
Código:

```
;procedimiento que convierte una cadena a mayuscula
        toUpperCase proc
        push bx

@@conv:  cmp byte ptr [bx],00h ;comparamos si la posicion de bx es null
        je @@fin   ;si es igual nos salimos
        cmp byte ptr [bx],20h ;comparamos si hay un espacio
        je @@espacio          ;si hay un espacio incrementamos 1 ya que el espacio no es letra
        sub byte ptr [bx],32;le restamos 32 para ajustar el valor de la minuscula a mayuscula
@@espacio:
        inc bx     ;incrementamos en 1 la posicion de la cadena de BX
        jmp @@conv ;seguimos convirtiendo los caracteres de la cadena
@@fin:
        pop bx
        ret
        endp
```

Procedimineto removeChar

Ejecución:



Código:

```
;-----------------------------PROCEDIMIENTOS-------------------------------------
;procedimineto que elimina las ocurrencias de una cadena de texto
;recibe la cadena por BX y elimina todas las ocurrencias indicadas por el registro AL
         removeCHAR proc
         push ax
         push bx
         push cx

@@comp:  mov cx,0              ;contador para regresar al indice adecuado
         cmp byte ptr [bx],al  ;comparamos la cadena con la ocurrencia a eliminar
         je @@del
         cmp byte ptr [bx],0
         je @@fin
         inc bx
         jmp @@comp
@@del:
         mov ah,byte ptr [bx+1] ;movemos en ah el siguiente valor de la cadena
         xchg byte ptr [bx],ah  ;intercambiamos los valores de posicion
         inc bx                 ;incrementamos bx
         inc cx                 ;para llevar el conteo del indice
         cmp byte ptr [bx],0    ;comparamos si [bx] es un cero
         je @@reg               ;si lo es regresamos a [bx] al indice inicial
         jmp @@del              ;si no seguimos moviendo las letras

@@reg:
@@index:
        dec bx                  ;decrementamos bx para que tenga su posicion inicial
        loop @@index
        jmp @@comp              ;brincamos a comparar para ver nuevas ocurrencias
        @@fin:                  ;fin del programa

         pop cx
         pop bx
         pop ax
         ret
```

Conclusión

Desarrollamos una serie de ejercicios para poder familiarizarnos con lo que es la creacion de procedimientos y el uso de los saltos y condiciones del ensamblador.