**Trabajo Práctico Final SPD GRUPO B**

Juego:“Ahorcado”

línea horizontal

# 

# 

# Integrantes:

*Lima Bruno Daniel* bdl1998@hotmail.com

*Pabón Hernan*  hermax.1022@gmail.com

*Arguello Gustavo Adrián*  gustavo.adrian.arguello@gmail.com

*Iozzia Ezequiel*  eze.iozzia@gmail.com

## Juego a Desarrollar

### Descripción

El “Ahorcado” en un juego clásico donde el jugador, para ganar una partida, tiene que adivinar una palabra que permanece oculta. Como pistas se le otorga la cantidad de letras que tiene esa palabra. Para adivinar, el jugador ira tecleando letras que él cree puedan estar contenidas en la palabra.



A medida que vaya acertando cada letras se ira mostrando las posiciones donde se encuentran dentro de la palabra oculta. Si el jugador nombra una letra que no pertenece a esa palabra, será penado por cometer un error, restándole una vida. Se puede equivocar como máximo de 5 letras nombradas, cuando desacierta la 6 letra (vida) el jugador pierde la partida. De caso contrario, el jugador ganara la partida unas ves que completa toda la palabra.



Para que el juego quede más pintoresco, es común que al cometer un error se muestre esa equivocación dibujando a un monigote o muñeco colgando de una soga. Se dibujara de a poco, con cada error se mostrar partes del muñeco. Si se completa el muñeco ahorcado significa que el jugador a llegado a sus 6 errores y ha perdido la partida.

### 



### Instrucciones de juego

Al iniciar el juego nos presenta el menú principal, donde podremos elegir entre 3 opciones (con el teclado)

* + **Start** :inicia una nueva partida, apriete cualquier tecla( salvo “s” o ”d”)
  + **Creditos** :muestra a los creadores del juego, apriete la tecla “s”
  + **Exit** :sale del juego, apriete la tecla “d”

**Start**: Una vez iniciada la partida el jugador tiene que ingresar una letra por teclado (indistinto si son mayúsculas o minúsculas), cada letra que el jugador vaya ingresando se irán guardando en la parte superior de la pantalla donde dice “Letras:”.

La palabra que se tiene que adivinar esta en el espacio con guiones que se muestra al lado del muñeco. Cada guion representa una letra, por ejemplo, si hay 5 guiones significa que la palabra tiene 5 letras. A medida que se ingresa las letras y se vaya adivinando cada una se irán mostrando las letras acertadas en esta línea.

En la parte inferior nos encontramos con la leyenda “VIDAS” que inicia con 6. Es un contador, por cada letra errada ingresada indicara cuantas letras nos queda por errar. Cuando ingresemos la 6 fallidas habremos perdido.

Al igual que las “VIDAS” el muñeco representa cuanto nos queda para perder. Con cada desacierto el muñeco se ira formando hasta terminar ahorcado.

Si hemos perdido todas las vidas el juego termina y nos mostrar la leyenda “game over” para continuar presione cualquier tecla. La acción anterior lo enviara al menú principal donde puede elegir una nueva partida.

**Créditos:** Una vez dentro de la opción para salir y volver al menú principal apriete cualquier tecla**.**

**Exit:** desde el menú principal, sale del juego.

### Compilación, Instalación y Ejecución

**Compilación/Instalación:**

En la carpeta donde esta “Tasm” de DOSBOX colocar los archivos:

* ***archivo.asm***
* ***grafico.asm***
* ***main.asm***
* ***palabras.txt***
* ***las imágenes contenidas en la carpeta “bmps”***

Para compilar, ejecutar las siguientes instrucciones en la consola de DOSBOX:

1. **tasm main.asm**
2. **tasm grafico.asm**
3. **tasm archivo.asm**
4. **tlink main.obj grafico.obj archivo.obj** (se puede omitir la terminación .obj de cada archivo)

**Ejecución:**

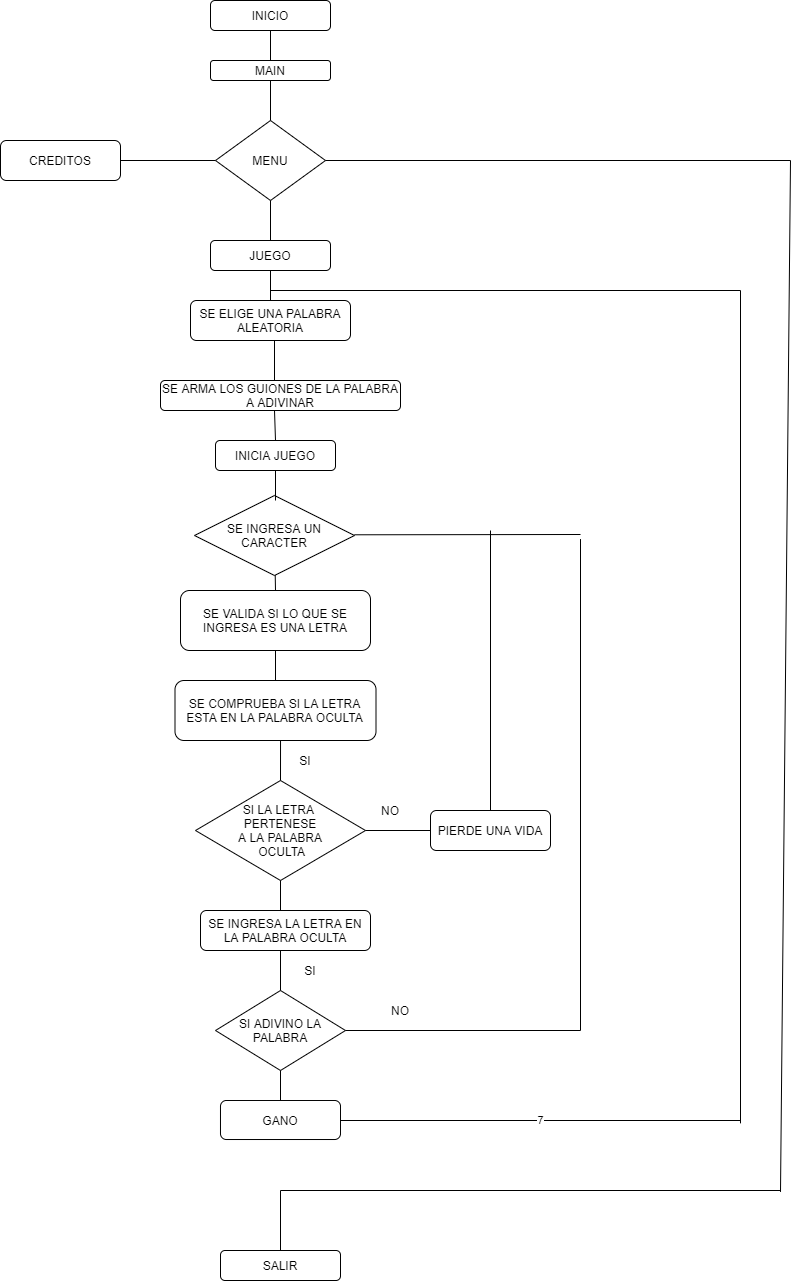
Para ejecutar escribir en la consola de DOSBOX:

* **main.exe** (o main a secas)

### 

### Diagrama de bloques

### 



### Funciones implementadas

El código se divide en 3 archivos: MAIN.ASM, ARCHIVO.ASM, GRAFICO.ASM. Cada uno con las siguientes funciones:

**MAIN.ASM:**

***menu proc:*** Recibe el offset de la imagen del menú en ‘dx’. Imprime el menú y valida la entrada.

***screen proc:*** Recibe en ‘Dx’ el offset de la imagen.bmp a imprimir junto los mensajes.

***minuscula proc:*** Evalúa un carácter que le llega en ‘AL’ y si es una letra mayúscula, la pasa a minúscula.

***l\_usadas proc:*** Guarda el carácter ingresado en el vector letra\_us. Recibe en ‘si’ el offset del vector.

***valida\_letra proc:*** Recibe el vector buscado en ‘di’ y el mostrado en ‘si’. Valida si el carácter ingresado está en la palabra buscada. Devuelve 01h en AH si está en la palabra y 00h en ‘Ah’ si no está. Actualiza el vector que se muestra.

***ganador proc:*** Recibe el vector buscado (DI) y el mostrado (SI). Compara y si son iguales devuelve 01h en AH (ganado), de lo contrario devuelve 00h en ‘AH’.

***limpia\_vec proc:*** Rellena un vector que recibió por parámetro, con un carácter también recibido por registro. Llega por SI el offset del vector a cargar, y en AX recibe dos parámetros. En AH el carácter con el que debe rellenar y en AL el tamaño del vector.

**ARCHIVO.ASM:**

***fun\_palabra proc:*** Recibe y envía por stack el offset de pal\_txt , lo recibe en SI para guardar la palabra leída. Durante la ejecución se hace uso de la función “fun\_aleatorio proc”, para obtener un número aleatorio que indicar la posición de la palabra del juego, que guardaremos en pal\_txt.

***fun\_aleatorio proc:*** Crea un número aleatorio entre cero y nueve. Devuelve el numero en ax(al).

**GRAFICO.ASM:**

***tam\_ad proc:*** Recibe el offset de pal\_ad en SI y pal\_txt en BX por stack. Ofuscada las letras (con guiones), para que el usuario des cubra la palabra. Devuelve la palabra ofuscada en el offset de pal\_ad en SI por stack.

***imprimePantalla proc:*** Entrada: DX = offset de la variable donde se almacena el nombre del archivo BMP que se quiere imprimir.

***cargaDelArchivo proc:*** Proceso 'cargaDelArchivo' se utiliza para abrir la imagen BMP y guardar el handler en [filehandler]. Entrada: DX = offset de la variable donde se almacena el nombre del archivo BMP que se quiere imprimir. Salida: [filehandle] con el handle del archivo bmp.

***lecturaDelHeader proc:*** Leemos el header del archivo BMP, son 54 bytes.El “filehandler” en BX y en DX el offset del “Header” .

***lecturaDelPalette proc:*** Leemos el palette de colores del archivo BMP, son 256 colores \* 4 bytes (400h), en cx. Y traemos el offset de Palette en DX.

***copiaDelPalette proc:*** Copiamos el palette de colores en la memoria de video por DX, el offset de Palette en SI. Copiamos el primer color en el puerto 3C8h y luego Copiamos el palette en el puerto 3C9h. Luego los colores rojo, verde, azul en el puerto y un carácter null después de cada color.

***copiaDelBitmap proc:*** Leemos línea por línea los gráficos (son 200 líneas en el formato VGA), mostrando las líneas desde abajo para arriba. Los gráficos BMP son guardados al revés. En DX nos llega el offset de ScrLine y copiamos la línea en la memoria de video usando SI.

***cierreDelArchivo proc:*** se utiliza para cerrar el archivo abierto indicado en filehandler, y dejando el handler libre para abrir otros archivos. En BX se manda el filehandle y se llama a la interrupción 21h para imprimir.