|  |  |
| --- | --- |
|  |  |

***Integrantes:*** Benítez, José Emanuel – Barrojo, Matías Ariel – Suarez, Juan Ignacio.

Conceptos teóricos utilizados de forma general

* *Los conceptos utilizados en los juegos MayMen, Adivinum, Warcoin y Ahorcado fueron:*

1. *Datos simples numéricos enteros y no numéricos boléanos.*
2. *Variables globales*
3. *Ciclos repetitivos: While, For.*
4. *Implementación de funciones.*
5. *Diferentes métodos de ordenamiento.*
6. *Arrays Unidimensionales y Bidimensionales.*

*Librería utilizada en los juegos*

*Se decidió crear una librería que sería para todos los juegos, dicha librería se llama “Owntools.h”, esta librería contiene las siguientes funciones de todos los juegos:*

*NOTA: Se utilizó la librería “conio.c” ya que nos facilita el trabajo para imprimir cuadros utilizando la función “gotoxy” a la cual se le envían coordenadas para que imprima en ese lugar un mensaje o símbolos. Y se utilizó la función “textcolor” o “textbackground” la cual nos permite cambiar el color a los textos o números que se mostraran en pantalla o cambiar el fondo.*

* *void TituloAV(); - Esta función genera el marco y el título del juego Adivinum mediante las funciones “gotoxy” (esta función nos facilita la impresión de marcos ya que nos permite desplazar el cursor hacia las coordenadas que le colocamos), que se las utiliza dentro de dos bucles FOR uno para el eje X y otro para el eje Y. Se los utiliza de esa forma ya que nos permite ir modificando el la coordenada y se imprima una línea o letra. Los gotoxy que están fuera de los bucle realizar una impresión en las esquinas de los cuadros que rodean al título Adivinum, Esta función también hace de utilidad a la función “textcolor”*
* *void MarcoResultadoAV(int x); - Esta función contiene las mismas funciones (las funciones gotoxy) para generar el marco del resultado de la partida, esta función recibe como parámetro que le es enviado desde la función “Resultado” del juego Adivunum 1 valor para cada caso, este valor recibido será verificado en la condición de si x es igual a 1, entonces generara los marcos que serán del ganador. Caso contrario se generara el marco del perdedor. Y regresa donde la llamaron lo cual fue en la función “Resultado”, y se le mostrara mensajes de felicitaciones y su puntaje en caso de que el usuario gane la partida caso contrario se le mostrara que perdió su puntaje 0 y el número que era de adivinar. Esta función también hace de utilidad a la función “textcolor”*
* *void InstruccionesAV(); - Esta función genera una tabla de asteriscos (\*) haciendo uso de la función “gotoxy” la cual encerrara a las instrucciones del juego Adivinum. Esta función también hace de utilidad a la función “textcolor”*
* *void TituloMYM(); - Esta función le pertenece al juego MayMen que realiza las mismas instrucciones que la función “TituloAV” anteriormente mencionada.*
* *void MarcoResultadoMYM(int PUNTAJE, int Numero)- En esta función recibe como parámetros el puntaje que obtuvo el usuario y el número a adivinar que es enviado desde la función “Inicio” del juego MayMen. Una vez en esta función pasara a una condición donde si el puntaje es mayor o igual a 1 se generara un cuadro mostrándole mensajes de felicitaciones y su puntaje obtenido en la partida, en caso contrario que no se cumpla la condición se le generara un cuadro mostrándole al usuario un mensaje que perdió la partida con su puntaje 0 y el número a adivinar. Ademas también los gotoxys que están fuera de los bucles son para imprimir las esquinas de los marcos y algunos para colocar los mensajes en sus lugares. Esta función también hace de utilidad a la función “textcolor”*
* *void TituloWC(); - Esta función le pertenece al juego Warcoin que realiza las mismas instrucciones que las funciones “TituloAV” y “TituloMYM” anteriormente mencionada.*
* *void MarcoResultadoWC(int TURNO)- Esta función tiene como parámetro el valor del turno que le se le envía desde la función “Juego” del juego Warcoin, dependiendo el valor que se le envié a esta función se ejecutara la condición donde si la variable ‘TURNO’ es igual a 1 entonces generara el marco del ganador y le muestra como mensaje al usuario que gano y su puntaje es 5. En caso contrario que la variable ‘TURNO’ no sea igual a 1 se generara el marco del perdedor y se le mostrara al usuario que la computadora gano y su puntaje es 0. Esta función también hace de utilidad a la función “textcolor”.*
* *void TituloAH(); - Esta función le pertenece al juego del Ahorcado donde se imprimirá las líneas y el título del mismo y también utiliza la función “textcolor”.*
* *void crearLineas(char símbolo, int cantidad); - Esta función es del juego Adivinum, que recibe los dos parámetros donde el primero es el símbolo que se ira mostrando y el segundo parámetro es la cantidad de veces que se va a repetir ese signo. Estos parámetros son enviados desde la función sin tipo “game” del juego del Ahorcado.*
* *void grafPersona(int numParte); - Esta función le pertenece al juego del Ahorcado recibe como parámetro una variable entera ‘numParte’ que es enviada desde la función “cuerpo” una vez en esta función se iniciara el bucle FOR que se ejecuta mientras que sea menor a la variable ‘numParte+1’, una vez dentro de este bucle se encuentran varias condición que dependiendo su valor verdadero se irán ejecutando y mostrando el grafico del muñeco utilizando un FOR con gotoxy. Luego que salga del bucle se ejecutaran otras condiciones en donde dependiendo de cuál se cumpla se hará un gotoxy para que el puntero vuelva a la sección para que el usuario ingrese la próxima letra a adivinar. Esta función también hace de utilidad a la función “textcolor”.*

*Aclaración: en la librería Owntools.h se implementaron las librerías “stdio.h”, “stdlib.h”, “conio.c”, “string.h”, “iostream”, “time.h”. Para que no sea necesario estar escribiendo esas librerías en cada archivo de los juegos y solamente se coloque la librería Owntools.h en ellos.*

*Análisis y desarrollo del Juego MayMen*

*Al comenzar el juego haciendo uso de la función “system()”para determinar el tamaño de la consola. Luego se iniciara el bucle White que se ejecutara mientras que la variable ‘op’ sea TRUE. Posterior a ello se llama a un subprograma llamado “NumeroGenerado” de tipo entero, el cual genera un número aleatorio entre 1 y 999, haciendo uso de la función SRAND (librería <time.h>). Una vez que este subprograma vuelve donde fue llamada se inicia otro subprograma llamado “Inicio” (función sin tipo), la cual contiene el cuerpo del juego.*

*Una vez dentro de la función, se iniciara el bucle FOR (el cual finalizara si adivinada el número, o terminado los 10 intentos), se le solicitara al usuario que ingrese un numero de 3 dígitos, el cual será comprobado si el numero está dentro rango propuesto, haciendo uso de una condición (IF). Si la condición no se cumple se le pedirá al usuario que ingrese nuevamente el número y no llevara penalización, en caso de que si se cumpla la condición se llamara a otra función sin tipo “rango” y actualizara el rango en el que se encuentra el número a adivinar. Una vez que la función “rango” regresa donde la llamaron. Se ira verificando las siguientes condiciones:*

* *Si el número ingresado es igual al número secreto, automáticamente sale del bucle, y se limpia la pantalla, luego llama a la función “MarcoResultadoMYM”.*
* *Si el número ingresado es mayor al número secreto, se le mostrar un mensaje indicando que el número secreto es menor con el rango en que se pueda encontrar el número secreto.*
* *Si el número ingresado es menor al número secreto, se le mostrara un mensaje indicando que el número secreto es mayor con el rango en que se puede encontrar el número secreto.*

*Si el usuario realizo los 10 intentos y no adivino el número, sale del bucle y realiza una limpieza de pantalla y se llama a la función “MarcoResultadoMYM”.*

*Una vez que el usuario gane o pierda en el juego, el subprograma, vuelve a la función main, y se le preguntara al usuario si “Quiere volver a jugar? (Para SI ingrese '1', para NO ingrese '2')”, la opción que elija el usuario se la guardara en una variable entera llamada ‘verificación’ y pasara a la condición donde si verificación es igual a 2 la variable booleana ‘op’ pasara de ser TRUE a FALSE y se le mostrar unos mensajes de despedida, en caso de ser falsa la condición, se limpiara la pantalla y el bucle se volverá a iniciar.*

*Análisis y desarrollo del Juego Adivinum*

*El juego comienza haciendo uso de la función “system()”para determinar el tamaño de la consola. Luego se inicia el bucle White que mientras la variables booleana ‘Opción’ sea TRUE se ejecutara. Una vez que el bucle se ejecuta llama a la función “generarNum” (función sin tipo, que tiene como parámetros el vector ‘container[4]’ y la variable entera donde se almacenara el numero generado de 4 cifras ‘y’). Esta función genera un numero el cual se verifica si es mayor a 0 posterior a eso dentro del bucle FOR se hace la búsqueda de si el numero generado está dentro del vector, si este no se encuentra se agrega al vector ‘container’, de lo contrario se resta un valor al contador y se vuelve a realizar el procedimiento hasta terminar el proceso del bucle FOR. Esto se realiza con el propósito de poder hacer uso de cada número de manera individual. Para que el número se muestre de manera de 4 cifras se realiza utilizando el bucle While que se ejecutara mientras que la variable ‘k’ (variable entera con valor 1) sea menor a 1001, cuando se ejecute el bucle, la variable entera ‘Num\_Generado’, almacenara el valor que genere realizando la operación : “‘Num\_Generado’=Num\_Generado+(container[t]\*k)”, después de realizar esa operación la variable ‘k’ se ira multiplicando por 10 y la variable ‘t’ ira sumando uno en uno hasta que la variable ‘k’ sea 1000. Una vez que sale del bucle la variable ‘Num\_Generado’ se almacena en la variable ‘Num’, luego la función “generarNum” regresa a la función “main”. Posteriormente se ejecuta la función “Inicio” la cual es el cuerpo del juego, en esta función se ejecuta el bucle FOR (el cual finaliza cuando el contador sea menor de 10), dentro de este bucle lo primero que se ejecutara es la función “TituloAV” (de la librería Owntools.h), luego se ejecuta la función “InstruccionesAV” (de la librería Owntools.h ). Después de ejecutarse esas funciones se le pedirá al usuario que ingrese el número. Una vez ingresado el número, se verificara en la condición si es que se encuentra entre 1000 y 9999. Si es que la condición no se cumple se le mostrara al usuario que vuelva a ingresar el valor y no tendrá penalización. Caso contrario que si se cumpla la condición se llamara a la función “verificarPosicion” (que tendrá como parámetro entero el arreglo container), la tarea de esta función es fragmentar el número para ir verificando la posición de cada digito. Luego ira verificando cuales de las condiciones se cumplen:*

* *Si el valor ingresado por el usuario es igual al número a adivinar, sale del bucle y se limpia la pantalla, luego de llama a la función sin tipo “Resultado” (de la librería Owntools.h) la cual le mostrara su puntaje*
* *Si el valor ingresado por el usuario no es igual al número a adivinar, se le mostrara un mensaje indicando la cantidad de número que son correctos y están en la posición correcta y la cantidad de números correctos pero que están en otra posición.*

*NOTA: Se le agrego una condición donde cada 2 intentos se limpie la pantalla.*

*En caso de que el usuario haga los 10 intentos y no adivine el número perderá la partida la cual la función sin tipo “Resultado” dentro de esta función, se declaró dos variables enteras ‘Ganador’ con valor 1 y la variable ‘Perdedor’ con valor 0, a estas variables se las usara más adelante. Después se verifica una condición de si el puntaje es mayor o igual a 1, llamara a otra función “MarcoResultadoAV” (que tiene como parámetro la variable que se declaró ‘Ganador’, y es una función de la librería Owntools.h) luego mostrara el puntaje que obtuvo el usuario. En caso de que no se cumpla esta condición también llama a la función “MarcoResultadoAV” (pero se le envía como parámetro la variable ‘Perdedor’), y una vez que vuelva a la función “Resultado” se le muestra su puntaje que seria 0 y el número a adivinar.*

*Luego de mostrar al usuario el resultado de la partida la función “Inicio” regresa a la función “main” donde fue llamada, y se le pregunta al usuario si ‘Quieres volver a jugar?(s/n)’, una vez que el usuario ingrese la opción que desea, se la almacenara en la variable char ‘Jugar’, después se utilizara la función “toupper” (que tiene como parámetro la variable ‘Jugar’) lo que hace esta función es tomar el carácter y devolverlo en versión en mayúscula. En caso de que el usuario haya ingresado la opción en minúsculo, de lo contrario no afectaría en nada. A continuación se verificaría si se cumple la condición donde si la variable ‘Jugar’ es igual a ‘N’ (NO) la variable boléana ‘Opción’ pasaría de TRUE a FALSE así mostrando dos mensajes de despedida y saliendo del bucle While. Y si no se cumple esa condición, ósea, si el usuario ingresa la letra ‘s’, se realiza una limpieza de pantalla y se vuelve a reiniciar la variable ‘Punt’ (Puntaje) y vuelve a ejecutarse el bucle While.*

*Análisis y desarrollo del Juego Warcoin*

*El juego comienza haciendo uso de la función “system()”para determinar el tamaño de la consola. Al comenzar, desde la función “main”, se inicia el ciclo While que se ejecutara mientras que la variable booleana ‘Op’ sea TRUE, entones se llama a la función “juego”, la cual contiene todo el juego. Al ingresar a la misma, se hace uso de la función sin tipo “PilaMoneda”, la cual genera el tamaño de la moneda. Luego se llama a la función “TituloWC” (de la librería Owntools.h) al finalizar dicho proceso, se continua a pedir datos al usuario (usando la función “PedirDatos”) que van a determinar los límites de extracción (mínimo y máximo).*

*Seguido a lo anterior, se genera el turno inicial (computadora o usuario) con la función ‘GenerarTurno’ y dicho valor se declara en la variable ‘turno’. Cuando esto se declara, da inicio un bucle While, el cual se ejecutara mientras que la variable ‘terminar’ sea distinto de 0. Una vez dentro del bucle se verificara una condición utilizando la variable ‘turno’ donde si es igual a 2 inicia la partida la computadora, mostrando primero la cantidad de monedas que hay en la pila. Luego en la variable ‘extracción’ se genera un número aleatorio (respetando los límites declarados), aprovechando la función ‘SRAND’, el cual será usado como extracción, a continuación se mostrara un mensaje con la cantidad de monedas que saco la computadora de la pila de monedas, y posteriormente hará el resta y luego pasara a una condición donde si pila es menor que el mínimo, se le mostrara la cantidad actual de monedas e pausara el juego por 2 segundo con la función “Sleep” y la variable ‘terminar’ pasara de valor 1 a 0, así saliendo del bucle While. En caso de que la condición no se cumpla lo que se hará es que a la variable ‘turno’ se le reste 1, cuando suceda eso la condición de si turno es igual 2 pasara a ser falsa y se ejecutara el turno del usuario mostrando la cantidad de monedas que le quedan a la pila luego se indicara que es el turno del usuario y se le pedirá que ‘Elija la cantidad de monedas a sacar’, se almacenara en la variable ‘moneda’. Una vez ingresada la cantidad se pasara a una condición donde si la ‘moneda’ es menor o igual al máximo y ‘moneda’ es mayor o igual al mínimo se hará la operación de restar esa cantidad a la pila de monedas, de lo contrario se le pedirá al usuario que ingrese el valor que este entre los valores mínimos y máximos.*

*Lo siguiente es otra condición donde si pila es menor a mínimo entonces muestra la cantidad de monedad en la pila y pausa por 2 segundo al juego con la misma función “Sleep” y la variable ‘terminar’ pasa de valor 1 a 0, así saliendo del bucle While. En caso contrario a la variable ‘turno’ se le suma 1 así pasando de nuevo a la condición para que empiece la computadora.*

*Dentro del While se utiliza la variable entera ‘contador’ inicializada en 0 que aumenta en 1 cada vez termina el turno del usuario o la computador. A esta variable se la utiliza en la condición de si contador es igual a 1, en la cual si es verdadera realiza un “system(“pause”)” y limpia la pantalla y se vuelve a inicializar en 0 a la variable ‘contador’ además vuelve a llamar a la función “TituloWC” y a la función “InformacionWC” (de la librería Owntools.h).*

*Una vez que sale del bucle While se limpia la pantalla y se llama a la función “MarcoResultadoWC” (de la librería Owntools.h) enviándole como parámetro la variable turno así cuando se ejecute sepa que resultado mostrar. Una vez mostrado el resultado vuelve a la función “Juego” y esta función vuelve donde fue llamada que es en la función “main”. Para que luego se le pregunte al usuario si ‘Quieres volver a jugar? (Para SI ingrese '1', para NO ingrese '2')’, la opción que elija el usuario se la guardara en una variable entera llamada ‘verificación’ y pasara a la condición donde si verificación es igual a 2, la variable booleana ‘op’ pasara de ser TRUE a FALSE y se le mostrar unos mensajes de despedida al usuario, en caso de ser falsa la condición, se limpiara la pantalla y el bucle se volverá a iniciar.*

*Análisis y desarrollo del Juego Ahorcado*

*Al comienzo, en la función “main”, se inicia utilizando “system()” para definir el tamaño de la consola. Posterior a ello, se ingresa a un bucle (WHILE) la cual su condición es mientras que la variable booleana ‘Opción’ sea TRUE se ejecutara. Una vez que se inicia el bucle lo primera que realiza es llamar a la función “start”, que es la pantalla de inicio del juego. En esta zona, al ingresar, se hace una limpieza de los arreglos de caracteres (word1 y word2), utilizando la función “vaciar”. Seguido de esto se inicia el ingreso de la primera palabra para el jugador A de parte del jugador B con la función “uploadWord”. En esta función, principalmente se le presenta las indicaciones de cómo debe hacerlo al ingreso (“letra por letra (después de cada letra presione ENTER)”) y la recomendación de palabras propuestas en las consigna del TP. Se utiliza un bucle (FOR), dentro de este, cada carácter ingresado se verifica si es alfabética utilizando una condición (IF) con la función booleana “isalpha()” (esta función comprueba si el carácter es alfabético). Si esto se cumple entonces es agregado al arreglo wordN (“N” es una referencia, ya sea 1 o 2) y agregando un valor a la variable “cantLetrasP[]”. De lo contrario, si no es alfabético, solicita un ingreso de nuevo en la misma posición. Cuando el usuario ingresa ‘0’, el bucle termina y con la función “strupr” se convierte en mayúscula todas las letras.*

*En regreso a la función “start”, se continua con la función “game”. Dicha función es la cual desarrolla el juego. Dentro de la misma, al comenzar, se llena la variable local “aux” con el uso de la función “rellenar” (rellena de guiones cada elemento del arreglo), la cantidad de guiones dependerá del valor de “num” (parámetro por valor de la variable “cantLetrasP[]”). Por consiguiente se ingresa a un bucle (WHILE) con la condición “i<11 and aciertos!=num”, donde aciertos se inicia en cero y a medida que el usuario acierta, este va aumentando. Dentro del bucle, se comienza mostrando el jugador que juega (A y luego B), numero de jugada, letras ya ingresadas (utilizando la función “mostrarLetras” la cual muestra los caracteres en el arreglo “letras”), la palabra que debe adivinar (impresión del arreglo “aux”), el estado del muñeco (el cual se hace uso del arreglo de tipo string y de pendiendo del valor de la variable “contCuerpo”, va a ir imprimiendo las partes del cuerpo). Luego llamara a la función “grafPersona” (de la librería Owntools.h), a continuación ira mostrar los datos de la situación del jugador, se solicita que ingrese una letra. La letra se deposita en la variable “letraInput”, la cual en un IF se verifica si es alfabética con el uso de la función booleana anteriormente nombrada (isalpha). Si esto no se cumple, me va a decir que vuelva a ingresar una letra sin penalizar al jugador. En la situación que el IF si se cumpla, se ingresa a otra condición, la cual haciendo uso de la función de tipo booleana “agrLetra”, la cual hace verificación de si la letra fue o no ingresada. Si el resultado es que la letra fue ingresada, te comenta sin penalización sobre dicha situación, pero si la letra no fue ingresada se procede a verificar si esta se encuentra dentro de la palabra propuesta por el contrincante con la función booleana “verificar”. Si pertenece a la misma, se hará el intercambio del guion por la letra en el lugar que se encuentra en el arreglo “aux” y se retornará el valor verdadero. En caso contrario, solo se retorna el valor falso. Estos valores booleanos se depositarán en la variable “valorResp”. La cual será sometida a evaluación en un IF que determinara la penalización aumentando un valor a “contCuerpo” y a “i”, si es que dicha variable posee valor “false”. Finalizado el bucle dado el hecho de que una de las condiciones se dejó de cumplir, se procede a calcular el puntaje con el siguiente IF, el cual determina que, si i es igual a 11, entonces el jugador recibe 0 puntos (esto es declarado en la variable puntaje (la cual trabaja como parámetro del arreglo “puntaje[0=jugador A y 1=jugador B]”)) y se muestra la palabra secreta.*

***Decisión de diseño***

*La decisión de cada diseño implementado en los diversos juegos, son logrados bajo la intención de facilitar la comprensión y desarrollo de jugabilidad de dichos programas, haciendo uso de un diseño ergonómico.*

*En cada juego se puede visualizar diferentes gamas de colores, pudiendo darle así un poco de carisma y volverlo atractivo a los ojos del usuario.*

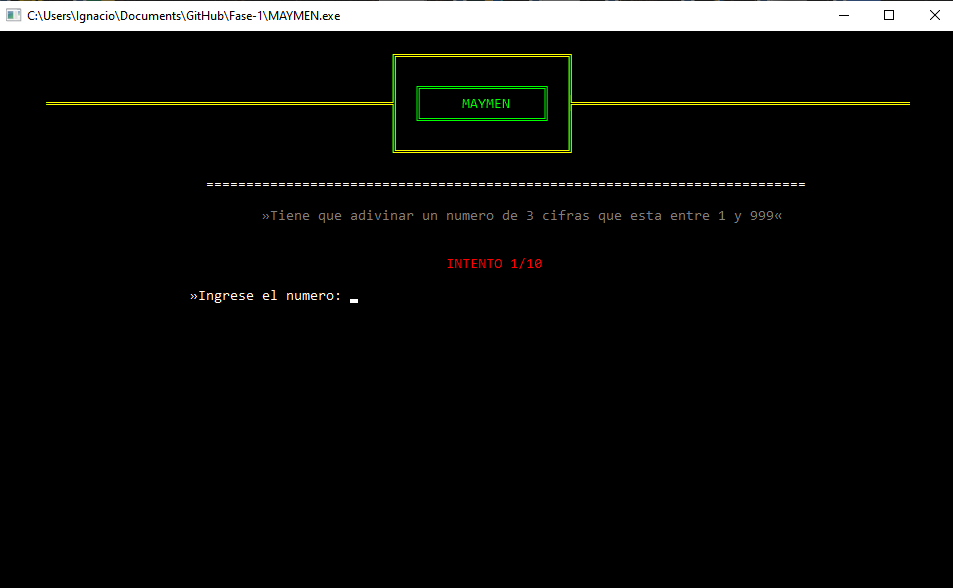
*Podemos presenciar títulos, los cuales se hacen presentes en todo momento. A tales títulos se les ha agregado diferentes caracteres que dan la sensación de un juego que se presenciaba en comienzo de la aparición de los mismos.*

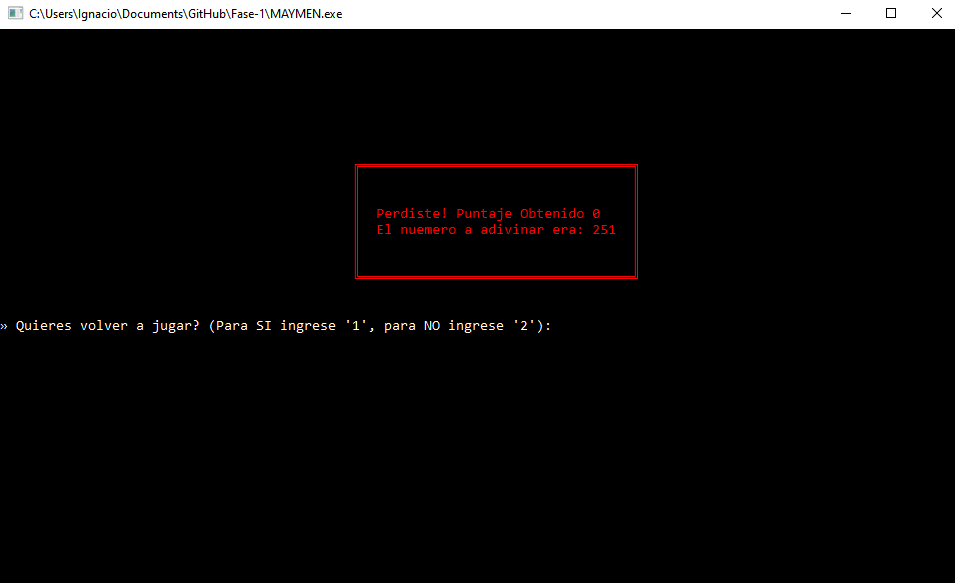
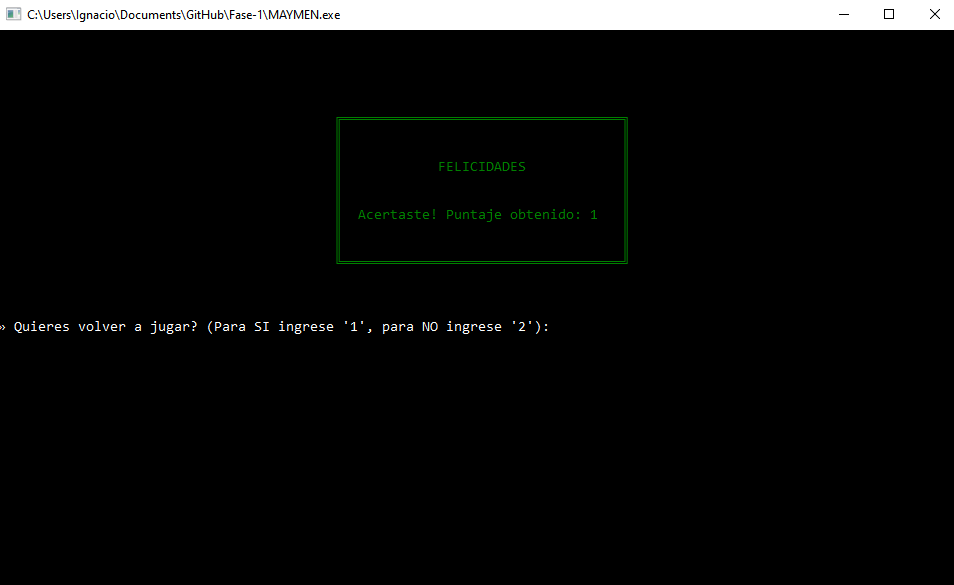
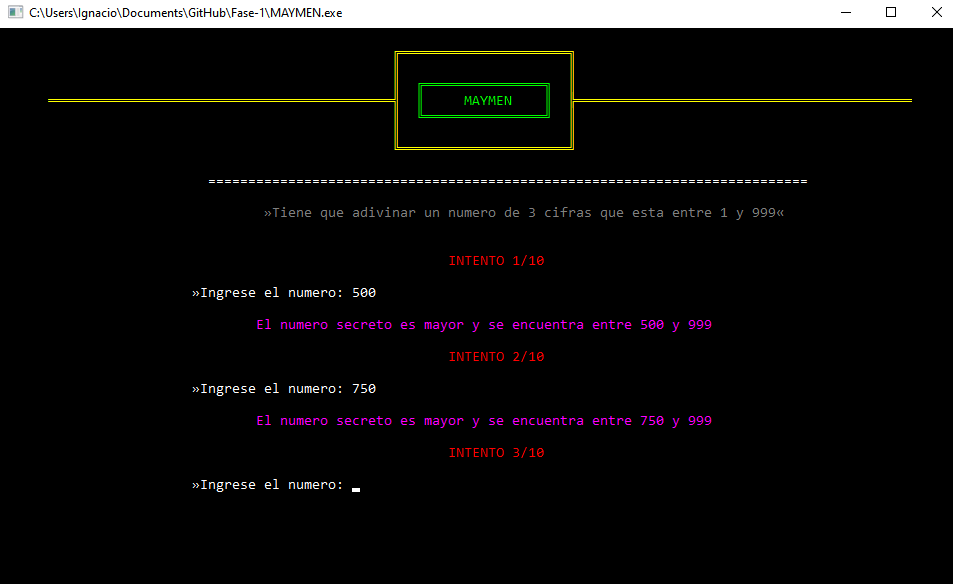
*También, se puede captar algunos carteles con información, otorgado las instrucciones al usuario, usando marcos simples hechos con variados caracteres.*

*Y por último vemos plasmado en el juego de letras "El Ahorcado", un gráfico que se presente por partes (basado en el diseño de 8 bits), el cual trata de poderle conceder al usuario, la fácil apreciación de la penalización de cada letra incorrecta ingresada.*

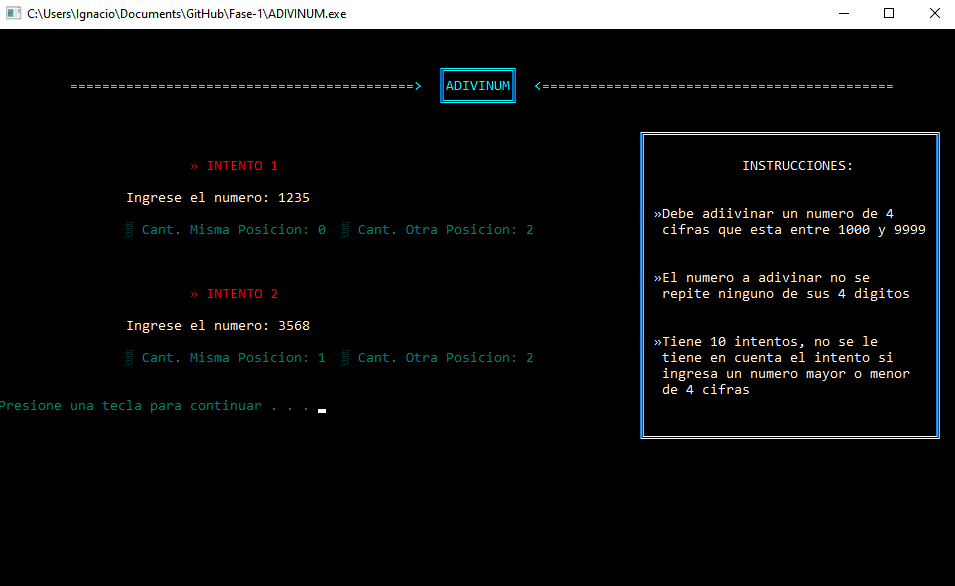
***Casos de prueba***

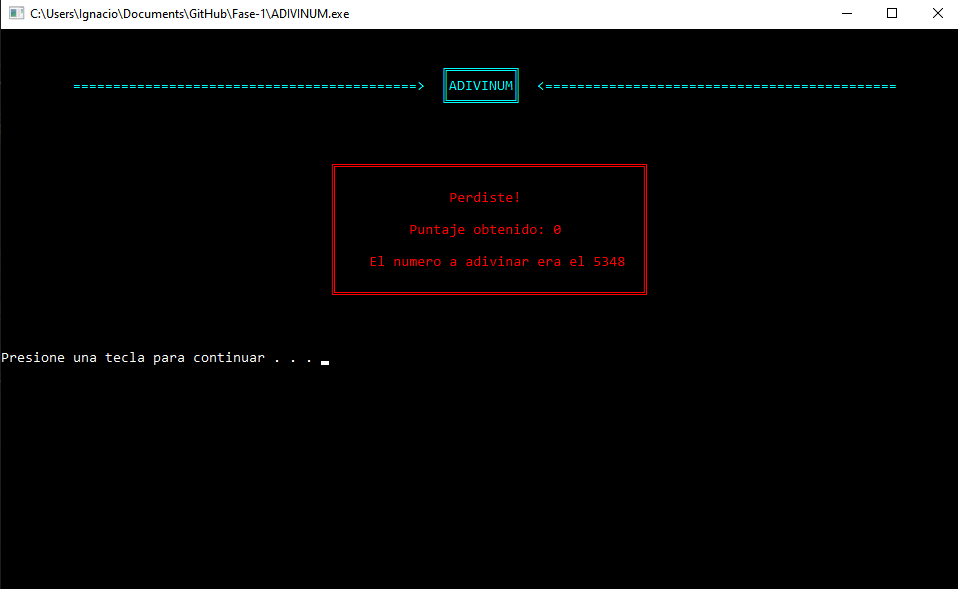
*MAYMEN*

**

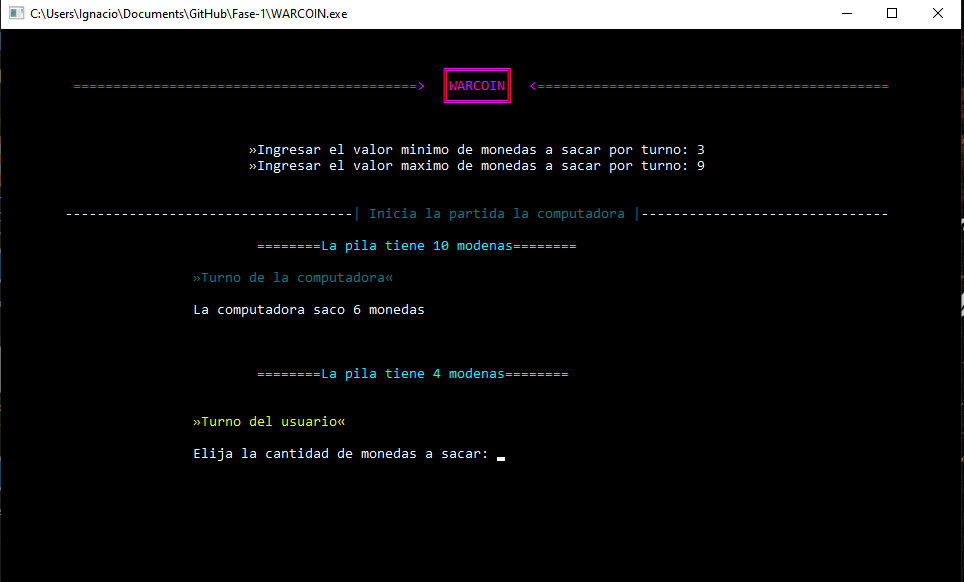


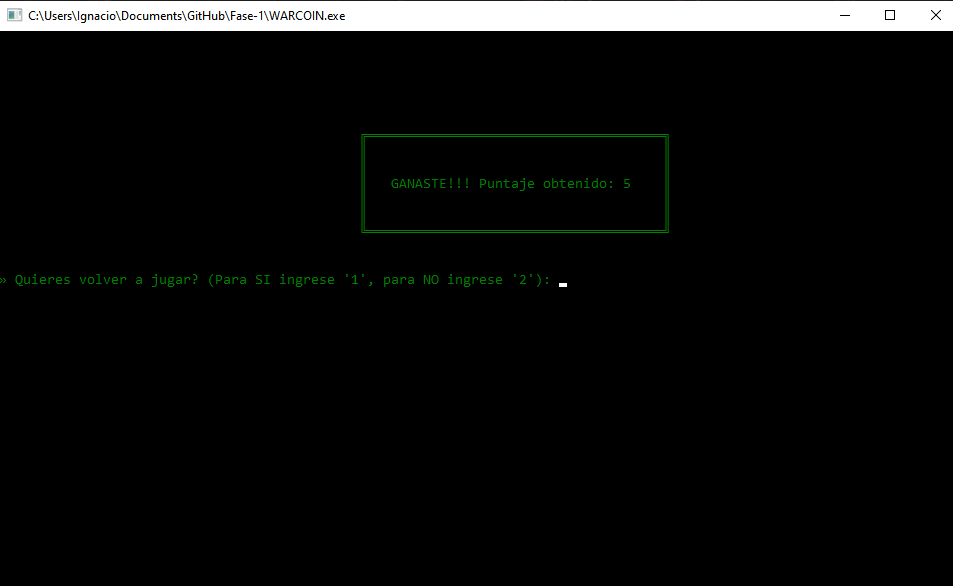
*ADIVINUM*

**



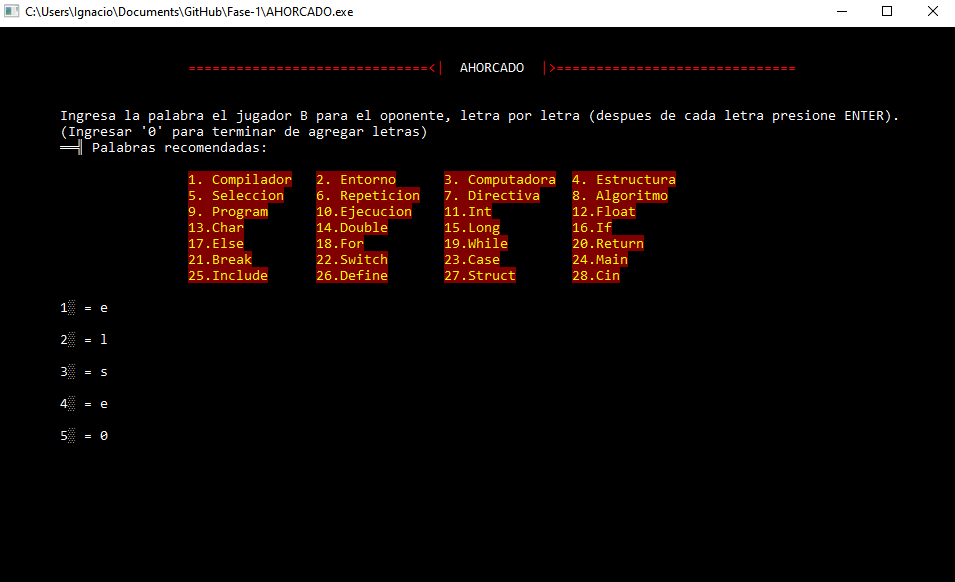
*WARCOIN*

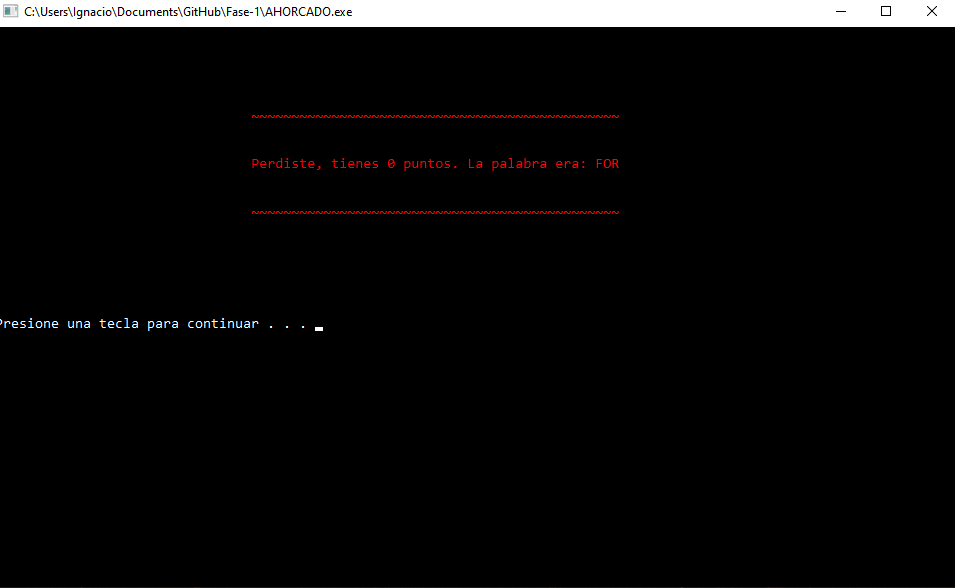
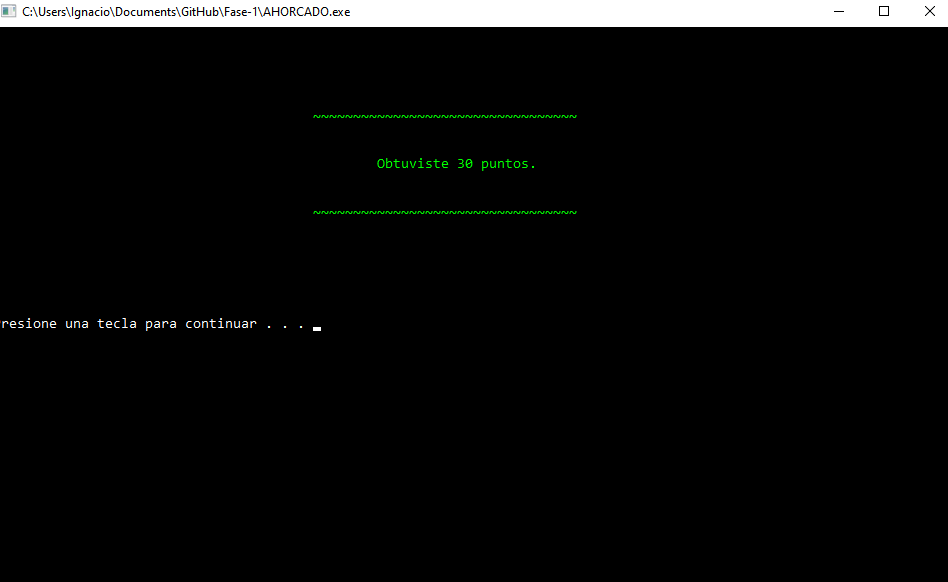
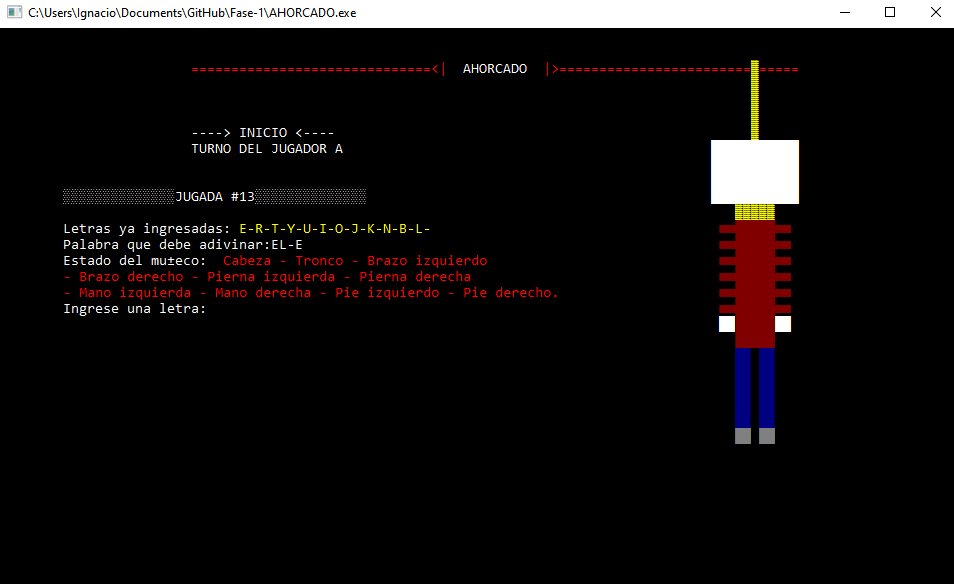
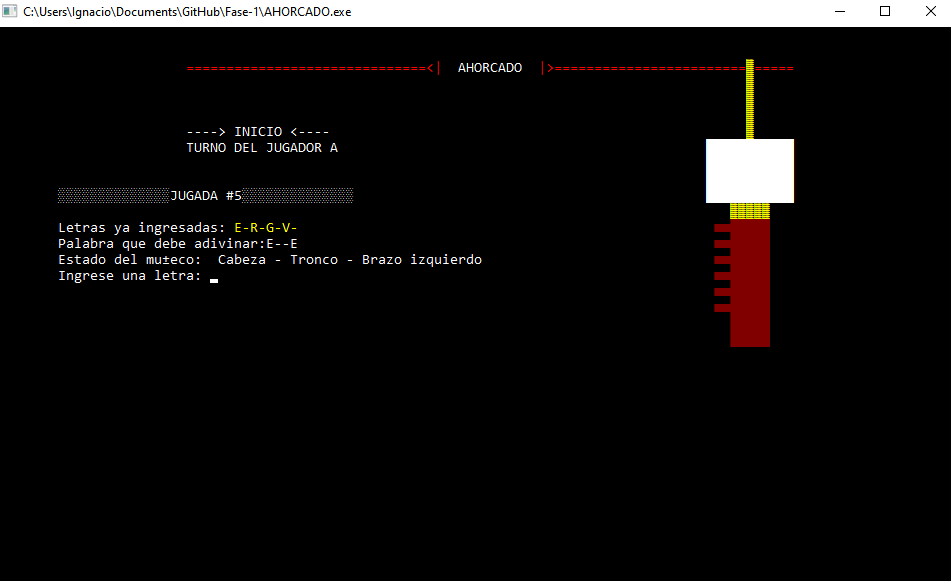
**

**



*AHORCADO*

**



***Experiencia***

*Al empezar con este proyecto, se pudieron reunir 2 integrantes de la localidad de Concepción, y a la vez hacían video llamada con el integrante que vive en otra localidad (Alberdi). Lo que se realizó en la primera reunión del grupo es que se habló sobre el tema de los horarios para llevar a cabo el proyecto de los juegos, ya que dado el hecho de las situaciones personales de cada integrante no se daban los tiempos. Aparte también se habló sobre los conocimientos de cada integrante para nivelarlos.*

*Pasaron unos días y la crisis sanitaria por el Covid-19, empeoro en la localidad de Concepción, lo cual evito nuestra reunión y se empezó a trabajar de forma virtual, haciendo uso de las herramientas: Zoom, Whatsapp y Github. A pesar de lo ocurrido, todo marcho de forma correcta, descartando los problemas sobre conectividad que sucedían de vez en cuando.*

*A pasar de los días e ir avanzando con el proyecto se nos presentaban dificultades en el juego de letras (Ahorcado), lo cual tuvimos que investigar sobre las variables tipos char, tipo string y todo lo que conlleve referente ello. Aparte se empezó a investigar sobre distintos tipos de funciones como gotoxy, textcolor, etc.*

*Luego de concluir todo el desarrollo de los juegos, concluimos en que el trabajo fue realizado sin problemas internos y pudimos llevar a cabo de manera eficiente el proyecto y el aprendizaje de nuevos conocimientos.*