Pantalla de computadora con un teclado de computadora

Descripción generada automáticamente con confianza mediaDOCUMENTACIÓN DEL 5QUILLO

En este documento trataremos de redactar las ideas y procedimientos que hemos llevado para completar el proyecto. Dividiremos el Word en 6 apartados: comprobaciones, menú, configuración, jugar, estadísticas y clasificación. También expondremos problemas que nos han podido surgir a la hora de programar y algunas de las fuentes de las que hemos obtenido la información.

Ejercicio realizado por los alumnos Jorge Cuadrado Criado y David Lavado González con dni (TU DNI) y 70910797A y correos (TU CORREO) y [david\_lavado@usal.es](mailto:david_lavado@usal.es) respectivamente

Comprobaciones previas:

Mediante la función comprobaciones hemos verificado que el jugador disponga de un archivo de configuración “config.cfg” (hablaremos de el más adelante), tenga permisos de lectura y escritura y además cerciorar que el archivo de configuración tenga el formato correcto.

Texto

Descripción generada automáticamenteAnte cualquier problema en un apartado de la función de comprobaciones, el problema específico será automáticamente reportado mediante un “echo”.

Creación del menú:

Texto

Descripción generada automáticamentePara el menú hemos recurrido a una secuencia de “echo” y un “case” para dirigir al jugador a la opción elegida. Dentro del case, existen 6 opciones, las principales que se acceden mediante la tecla indicada ya sea mayúscula o minúscula como jugar, salir… Y por último el que caso que recoge cualquier tecla que no sea la asignada a un modo, en este caso nos sacará por pantalla que escojamos una tecla válida.

Texto

Descripción generada automáticamente

Configuración:

En este apartado podremos consultar la configuración inicial que vendrá dada por el archivo “config.cfg” y cambiar la misma. En el archivo “config.cfg” se indicará el número de jugadores, estrategia y ruta al log inicial. En la función, tras elegir si deseamos cambiar un ajuste o no, se leerá por teclado la nueva configuración, en caso de no ser viable, se mandará por pantalla la razón por la cual no es viable.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Jugar:

Este apartado es el más largo por lo que lo dividiremos en funciones para explicar más profundamente la idea que hemos llevado para realizar el trabajo.

-Crear la baraja es lo primero que hemos realizado mediante 3 “arrays”, una para los palos, otro para el número de las cartas y otro para la baraja. Después, gracias a el uso consecutivo de bucles “for” para rellenar la baraja.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

-Como en cualquier juego de cartas, ha sido necesario barajar las cartas para conseguir que cada jugador reciba unas cartas aleatorias. Para ello hemos recurrido a un bucle “for” de 40 iteraciones de tal manera que dos índices, que representan 2 cartas cuales quiera dentro de la baraja, sean intercambiados entre sí mediante la función “random”.

Texto

Descripción generada automáticamente

-Después de crear y barajar las cartas, a través de una función las hemos repartido a cada jugador, mediante un “case” para las diferentes cantidades de jugadores y después mediante un simple “for” y un “do” para repartirlas.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

-Una vez los jugadores tengan ya sus cartas, hemos creado 4 “arrays” que suponen el numero completo de cartas para cada palo y después hemos hecho un tablero en el que se muestran las cartas de cada jugador y el número completo de cartas para ir marcándolas según hayan sido utilizadas por los jugadores.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

-En este juego comienza el jugador que tenga el 5 de oros, nosotros hemos creado una función para asegurarnos de ello.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

-Tras sacar el 5 de oros, empieza la partida y esta es la parte más extensa por la que nos limitaremos a explicar brevemente nuestra idea ya que si podemos una captura de todas las líneas de este apartado de código se haría muy largo y pesado leer y entender esta documentación.

Lo primero era controlar los turnos ya que variaba dependiendo del número de jugadores.



Texto

Descripción generada automáticamente

Una vez instanciados la secuencia de turnos queda realizar el turno del jugador y de la máquina. La forma en la que funciona el turno de la máquina variará dependiendo de la estrategia que estemos jugando.

Texto

Descripción generada automáticamente

Esta imagen muestra una pequeña parte del funcionamiento de la máquina, hemos recurrido a una función genérica para aportar simpleza y limpieza al código en el que se comprueba secuencialmente que carta puede lanzar. Si estamos en la estrategia 1, la primera carta que sea posible lanzarse será la utilizada por la máquina. En la estrategia 2, se busca la carta a lanzar mediante el algoritmo de la burbuja y es lanzada. En ambas estrategias habría que comprobar previamente si la máquina tiene cartas válidas para lanzar y en caso contrario saltar turno. Todas las comprobaciones para las cartas válidas es importante destacar que se realizan para cada uno de los 4 palos.

Texto

Descripción generada automáticamente

En esta imagen se muestra una pequeña parte del código utilizado para el turno del jugador. La particularidad del turno del jugador respecto al de la máquina es que el usuario es el que elige la carta a sacar, es decir, podrá elegir cartas que no sea posible usar también. Si el jugador elige una carta no válida nos hemos asegurado de que le sea informado que la carta elegida no es posible lanzarse. Puede existir también la posibilidad de que el jugador no posea en su mano ninguna carta válida para lanzarse en un turno, en ese caso, hemos decidido que lo mejor sería una comparación previa de si el jugador tiene cartas válidas, en el caso de no tener, su turno será saltado reportando por pantalla la razón automáticamente.

-Por último, al finalizar la partida, teníamos que guardar algunos datos en un fichero sin sobrescribir los de las partidas anteriores. Para ello hemos creado cada variable que deseamos pasar al fichero y le hemos dado la información recopilada durante el juego, finalmente con un “echo” de las variables al fichero hemos transferido la información.

Texto

Descripción generada automáticamente

Estadísticas:

Este apartado lo hemos resuelto mediante una función a la que hemos llamado “estadísticas”. Al principio de esta se comprueba primero si existe en el LOG y en el caso de hacerlo, si está vacío ya que si no hay nada en el LOG no existe estadísticas las cual mirar. En ese caso será reportado por pantalla que es necesario jugar al menos una partida para comprobar las estadísticas. El número total de líneas en el archivo LOG será el número de partidas jugadas ya que cada partida implica una línea, para ello mediante la sentencia “grep -c” hemos obtenido el número de líneas para poder indicar el número de partidas jugadas. Mediante el uso de variables y cálculos sencillos hemos sido capaces de mostrar y almacenar todas las estadísticas solicitadas. Aspectos individuales como puntos por jugador o victorias hemos usado bucles “for” para almacenar al final de cada partida esa información.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Clasificación:

Este apartado es muy parecido al anterior, al principio se verifica la existencia del LOG y si está vacío se reporta por pantalla el hecho de que es necesario una partida como mínimo para establecer una clasificación. El resto es también muy similar a las estadísticas, se recorre el archivo buscando la información deseada y es guardada en la variable que será mostrada por pantalla.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Problemas a la hora de crear el programa:

Afrontar un lenguaje nuevo siempre es difícil y esta vez no iba a ser menos, muchas de las sentencias son prácticamente iguales a las de otros lenguajes, pero también había algunas nuevas a las que hemos tenido que acostumbrarnos. A pesar de ello, gracias a las sesiones de prácticas subidas en avellano y algunos vídeos y páginas de internet hemos conseguido afrontarlo bien. Enfocándonos ya más en el programa como tal, hemos tenidos algunas dificultades:

-La disposición de los turnos fue algo compleja debido a que en cada ronda hay un número de turnos igual al número de jugadores, es decir si hay 2 jugadores, habrá 2 turnos por ronda. Debido a esto ocurría un problema ya que mediante nuestra manera de plantear los turnos teníamos que volver al primer jugador de alguna manera. Hemos resuelto este problema mediante la condición de que cuando el turno fuera el del último jugador, se reiniciaran los turnos, es decir, vuelve al primero.

-Hemos tenido que crear una función extra llamada “puso\_carta” ya que en el turno de la máquina se ponían todas las cartas posibles en vez de una por turno. Con esta función nos hemos asegurado de que la máquina usara una sola carta por turno a la vez que hemos dado robustez y limpieza al código.

-La baraja española tiene como particularidad que se pasa del número 7 al 10 por lo que ha complicado las cosas a la hora de ordenar las cartas debido a que con una comprobación genérica para ver que carta va después de cada una, la siguiente al 7 sería el 8 y por lo tanto ocurría un error. Hemos resuelto este problema añadiendo una condición extra de tal manera que contemple el número 10 a la hora de colocar una carta.

Aspectos positivos y a destacar de nuestro programa:

-Hemos querido hacer énfasis en la capacidad de poder modificar cualquier aspecto del fichero “config.cfg” individualmente, de tal manera que mediante una correcta utilización del menú de opciones sería posible cambiar por ejemplo los jugadores de la partida sin alterar nada más.

-Para nosotros ha sido una gran prioridad el crear un programa robusto de tal manera que el cualquier comportamiento inusual o mínimamente diferente al esperado por el programa será reportado por pantalla al usuario. Para buscar esta robustez hemos llevado a cabo numerosas sentencias condicionales para asegurar que el usuario posea todos los elementos necesarios para jugar, es decir, que el fichero “config.cfg” exista y no esté vacío, que la ruta del LOG sea válida, que el número de jugadores sea factible…

-Un aspecto muy importante era asegurarnos de que no hubiera huecos por donde romper el programa ya que el usuario muchas veces no usa los programas de la manera que es debido pudiendo crear problemas y romper todo. De esta manera, en el turno del jugador nos hemos cerciorado de que el numero de la carta escogido sea un número y no una letra o un carácter especial y que la carta escogida sea válida.