SISTEMAS OPERATIVOS I 2º DE G.I.I.

Primera practica

5illo.sh

Primer miembro: JORGE CUADRADO CRIADO (i0924801)

Segundo miembro: DAVID LAVADO GONZALEZ (i0910797A)

1 de noviembre de 2023

UNIVERSIDAD DE SALAMANCA

**INICIO DEL PROGRAMA**

El programa opera en dos modos. Cuando se ejecuta con el argumento -g, muestra documentación con los nombres de los autores y detalles sobre tres estrategias utilizadas. Después, espera una entrada del usuario para continuar. Si se ejecuta sin -g o con cualquier otro argumento, muestra un mensaje de error y se detiene. El propósito principal es proporcionar información sobre el programa y las estrategias.

if [ "*$1*" = "-g" ]

then

    echo "      =========================="

    echo "              DESCRIPCION       "

    echo "      =========================="

    echo

    echo "               AUTORES"

    echo "      ->Jorge Cuadrado Criado"

    echo "      ->David Lavado González"

    echo

    echo "      =========================="

    echo "              ESTRATEGIAS       "

    echo "      =========================="

    echo

    echo "      Estrategia 0 Aleatorio: Los jugadores colocan automáticamente la primera carta que puedan colocar"

    echo "      Estrategia 1 Mayores: Los jugadores colocan la carta de mayor valor que puedan colocar para reducir el número de puntos restantes al final de la partida"

    echo "      Estrategia 2 Menores: Los jugadores colocan la carta de menor valor que puedan colocar"

    echo

    read -p "Pulse una tecla para continuar..."

    exit 0

fi

*#Si la funcion se invoca con un parametro diferente a -g se avisa que solo puede usar -g*

if [[ "*$1*" != "" && "*$1*" != "-g" ]]

then

    echo "Solo se puede usar -g"

    exit 1

fi

**COMPROBACIONES PREVIAS:**

Mediante la función comprobaciones\_cfg 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.

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

A screenshot of a computer code

Description automatically generated

**A screen shot of a computer program

Description automatically generated**

La función Comprobaciones\_log se asegura de que el archivo de registro especificado en $LOG exista y tenga los permisos adecuados de lectura y escritura antes de continuar con la ejecución del programa. Si alguna comprobación falla, muestra un mensaje de error y, en el caso de permisos insuficientes, finaliza el programa con un código de error -1.

**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.

A screen shot of a computer program

Description automatically generated

Este código crea un menú interactivo en un bucle infinito. El usuario puede seleccionar diferentes opciones (configuración, juego, estadísticas, clasificación o salir) y el programa realizará la acción correspondiente. Si se elige una opción incorrecta, se muestra un mensaje de error y se vuelve a mostrar el menú.

El bucle continuará hasta que el usuario elija salir.

**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 (pudiendo elegir si está en el mismo directorio o no). 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

A screenshot of a computer program

Description automatically generated

**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 al uso consecutivo de bucles “for each” 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” asignando a cada jugador el numero de cartas que le corresponden según el numero de jugadores en su array y almacenando en una variable cuantas cartas tiene cada uno para su posterior uso, además en caso de ser 3 jugadores el bucle se recorre con normalidad y al final la carta sobrante se otorga al usuario.

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 para ello usamos la función printf() para poder dar formato y encolumnar cada array.

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. Para ello recorremos todos los arrays de los jugadores y una vez encontrado el “5 de oros” este se cambia por vacío (“”) en su mano y se cambia el valor en el tablero añadiendo un (+) que representa si una carta esta colocada.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Lo primero era controlar los turnos ya que variaba dependiendo del número de jugadores. En esta función se asigna unas variables genéricas los datos del jugador en su turno es decir su mano, su turno y el numero de cartas para poder pasar estos valores a la función de selección de carta de manera genérica y no tener que hacer una por turno. Pero primero que debemos hacer antes de comentar con los turnos es llamar a la función comprobar vacíos que explicaré más adelante esta determina según un valor de 0, 1, 2, 3 y 4 si las manos de los jugadores están vacías en caso de estarlo se pone la variable **VACIO** al número del jugador que no tiene cartas o cero si todos tienen cartas, esto es nuestra condición de victoria.

function juego {

    SECONDS=0

    clear

    crear\_baraja

    barajar

    clear

    repartir\_por\_jugadores

    mostrar\_tablero

    buscar\_5oros

    comprobar\_vacios

while [ "$VACIO" == "0" ];

do

    comprobar\_vacios

    if [ ! "$VACIO" == "0" ]; then

        break

    else

        CONTADOR\_TURNOS=$((CONTADOR\_TURNOS + 1))

*# Si el turno es mayor que el número de jugadores, se reinicia el turno*

        if [ $TURNO -gt $JUGADORES ]; then

            TURNO=1

        fi

        case $TURNO in

            1)

                echo

                echo "------------> TURNO DEL JUGADOR $TURNO <------------"

                echo

                turno\_usuario

                TURNO=2

                comprobar\_vacios

                ;;

            2)

                echo

                echo "------------> TURNO DEL JUGADOR $TURNO <------------"

                echo

                NUMERO\_CARTAS\_TOTAL=$NUMERO\_CARTAS\_JUGADOR2

                CARTAS\_JUGADOR=("${JUGADOR2[@]}")

                TURNO\_JUGADOR=$TURNO

                turno\_maquina

                comprobar\_vacios

                ;;

            3)

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.

function turno\_maquina {

case $ESTRATEGIA in

    0)

        seleccion\_secuencial

    ;;

    1)

    for ((k=0;k<$JUGADORES;k++)); do

        for ((i = 0; i < ${#CARTAS\_JUGADOR[@]}; i++)); do

            for ((j = i + 1; j < ${#CARTAS\_JUGADOR[@]}; j++)); do

                NUMERO=$(echo "${CARTAS\_JUGADOR[i]}" | cut -d " " -f 1)

                NUMERO\_SIG=$(echo "${CARTAS\_JUGADOR[j]}" | cut -d " " -f 1)

                if [ -n "$NUMERO" ] && [ -n "$NUMERO\_SIG" ] && [ "$NUMERO" -lt "$NUMERO\_SIG" ]; then

                  TEMP="${CARTAS\_JUGADOR[i]}"

                  CARTAS\_JUGADOR[i]="${CARTAS\_JUGADOR[j]}"

                  CARTAS\_JUGADOR[j]="$TEMP"

                fi

            done

        done

        case $TURNO in

            2)

                JUGADOR2=("${CARTAS\_JUGADOR[@]}")

                ;;

            3)

                JUGADOR3=("${CARTAS\_JUGADOR[@]}")

                ;;

            4)

                JUGADOR4=("${CARTAS\_JUGADOR[@]}")

                ;;

        esac

    done

        seleccion\_secuencial

    ;;

    2)

    hecho=$((JUGADORES - 1))

    while [ $hecho -ne 0 ]; do

        for ((i = 0; i < ${#CARTAS\_JUGADOR[@]}; i++)); do

            for ((j = i + 1; j < ${#CARTAS\_JUGADOR[@]}; j++)); do

                NUMERO=$(echo "${CARTAS\_JUGADOR[i]}" | cut -d " " -f 1)

                NUMERO\_SIG=$(echo "${CARTAS\_JUGADOR[j]}" | cut -d " " -f 1)

                if [ -n "$NUMERO" ] && [ -n "$NUMERO\_SIG" ] && [ "$NUMERO" -gt "$NUMERO\_SIG" ]; then

                  TEMP="${CARTAS\_JUGADOR[i]}"

                  CARTAS\_JUGADOR[i]="${CARTAS\_JUGADOR[j]}"

                  CARTAS\_JUGADOR[j]="$TEMP"

                fi

            done

            case $TURNO in

                2)

                    JUGADOR2=("${CARTAS\_JUGADOR[@]}")

                    ;;

                3)

                    JUGADOR3=("${CARTAS\_JUGADOR[@]}")

                    ;;

                4)

                    JUGADOR4=("${CARTAS\_JUGADOR[@]}")

                    ;;

            esac

        done

        hecho=$((hecho - 1))

        break

    done

        seleccion\_secuencial

    ;;

    \*)

        echo "ERROR: ESTRATEGIA NO VÁLIDA"

    ;;

esac

}

Si la estrategia es 0 se llama automáticamente a la función de selección secuencial (juego aleatorio) que explicaremos mas adelante, si es 1 o 2 el objetivo es que la maquina juegue las cartas mayores o menores primero respectivamente para ello hemos querido mantener el algoritmo de búsqueda secuencial pero antes con un algoritmo de burbuja hemos ordenado de menor a mayor o de mayor a menor las cartas de forma que al recorrer el array la primera carta posible que puedan poner será o la mas grande o la más pequeña.

function seleccion\_secuencial

{

    for ((i = 0; i < $NUMERO\_CARTAS\_TOTAL && $TURNO\_JUGADOR==$TURNO; i++)); do

        if [ ! -z "${CARTAS\_JUGADOR[i]}" ]; then

            CARTA="${CARTAS\_JUGADOR[i]}"

            PALO=$(echo "$CARTA" | cut -d " " -f 3)

            NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

            if [ "$NUMERO" == 5 ]; then

                PUSO\_CARTA=false

                case $PALO in

                    "bastos")

                        BASTOS[4]="(+) 5 de bastos"

                        echo "JUGADOR $TURNO PONE CARTA: 5 de bastos"

                        CARTAS\_JUGADOR[i]=""

                        case $TURNO in

                            2)

                                JUGADOR2[i]=""

                                ;;

                            3)

                                JUGADOR3[i]=""

                                ;;

                            4)

                                JUGADOR4[i]=""

                                ;;

                        esac

                        TURNO=$((TURNO + 1))

                        PUSO\_CARTA=true

*#clear*

                        mostrar\_tablero

                        break

                        ;;

                    "copas")

                        COPAS[4]="(+) 5 de copas"

                        echo "JUGADOR $TURNO PONE CARTA: 5 de copas"

                        case $TURNO in

                            2)

                                JUGADOR2[i]=""

                                ;;

                            3)

                                JUGADOR3[i]=""

                                ;;

                            4)

                                JUGADOR4[i]=""

                                ;;

                        esac

                        TURNO=$((TURNO + 1))

                        PUSO\_CARTA=true

*#clear*

                        mostrar\_tablero

                        break

                        ;;

                    "espadas")

                        ESPADAS[4]="(+) 5 de espadas"

                        echo "JUGADOR $TURNO PONE CARTA: 5 de espadas"

                        case $TURNO in

                            2)

                                JUGADOR2[i]=""

                                ;;

                            3)

                                JUGADOR3[i]=""

                                ;;

                            4)

                                JUGADOR4[i]=""

                                ;;

                        esac

                        TURNO=$((TURNO + 1))

                        PUSO\_CARTA=true

*#clear*

                        mostrar\_tablero

                        break

                        ;;

                esac

            else

                PUSO\_CARTA=false

                case $PALO in

                    "bastos")

                        for ((j = 0; j < 10; j++)); do

                            if [[ "$CARTA" == "${BASTOS[j]}" ]]; then

                            if [[ "$NUMERO" == "10" ]]; then

                                if [[ "${BASTOS[j-1]}" == "(+) 7 de bastos" ]]; then

                                    BASTOS[j]="(+) ${NUMERO} de bastos"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de bastos"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                        2)

                                            JUGADOR2[i]=""

                                            ;;

                                        3)

                                            JUGADOR3[i]=""

                                            ;;

                                        4)

                                            JUGADOR4[i]=""

                                            ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

elif [[ "${BASTOS[j+1]}" == "(+) $((NUMERO + 1)) de bastos" ]] || [[ "${BASTOS[j-1]}" == "(+) $((NUMERO - 1)) de bastos" ]] ; then

                                    BASTOS[j]="(+) ${NUMERO} de bastos"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de bastos"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                        2)

                                            JUGADOR2[i]=""

                                            ;;

                                        3)

                                            JUGADOR3[i]=""

                                            ;;

                                        4)

                                            JUGADOR4[i]=""

                                            ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

                            fi

                    done

                        ;;

                    "copas")

                        for ((j = 0; j < 10; j++)); do

                            if [[ "$CARTA" == "${COPAS[j]}" ]]; then

                            if [[ "$NUMERO" == "10" ]]; then

                                if [[ "${COPAS[j-1]}" == "(+) 7 de copas" ]]; then

                                    COPAS[j]="(+) ${NUMERO} de copas"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de copas"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                        2)

                                            JUGADOR2[i]=""

                                            ;;

                                        3)

                                            JUGADOR3[i]=""

                                            ;;

                                        4)

                                            JUGADOR4[i]=""

                                            ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

elif [[ "${COPAS[j+1]}" == "(+) $((NUMERO + 1)) de copas" ]] || [[ "${COPAS[j-1]}" == "(+) $((NUMERO - 1)) de copas" ]] ; then

                                    COPAS[j]="(+) ${NUMERO} de copas"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de copas"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                    2)

                                        JUGADOR2[i]=""

                                        ;;

                                    3)

                                        JUGADOR3[i]=""

                                        ;;

                                    4)

                                        JUGADOR4[i]=""

                                        ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

                        fi

                    done

                        ;;

                    "espadas")

                        for ((j = 0; j < 10; j++)); do

                            if [[ "$CARTA" == "${ESPADAS[j]}" ]]; then

                            if [[ "$NUMERO" == "10" ]]; then

                                if [[ "${ESPADAS[j-1]}" == "(+) 7 de espadas" ]]; then

                                    ESPADAS[j]="(+) ${NUMERO} de espadas"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de espadas"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                        2)

                                            JUGADOR2[i]=""

                                            ;;

                                        3)

                                            JUGADOR3[i]=""

                                            ;;

                                        4)

                                            JUGADOR4[i]=""

                                            ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

elif [[ "${ESPADAS[j+1]}" == "(+) $((NUMERO + 1)) de espadas" ]] || [[ "${ESPADAS[j-1]}" == "(+) $((NUMERO - 1)) de espadas" ]] ; then

                                    ESPADAS[j]="(+) ${NUMERO} de espadas"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de espadas"

                                    CARTAS\_JUGADOR[i]=""

                                case $TURNO in

                                    2)

                                        JUGADOR2[i]=""

                                        ;;

                                    3)

                                        JUGADOR3[i]=""

                                        ;;

                                    4)

                                        JUGADOR4[i]=""

                                        ;;

                                esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

                            fi

                    done

                        ;;

                    "oros")

                        for ((j = 0; j < 10; j++)); do

                            if [[ "$CARTA" == "${OROS[j]}" ]]; then

                            if [[ "$NUMERO" == "10" ]]; then

                                if [[ "${OROS[j-1]}" == "(+) 7 de oros" ]]; then

                                    OROS[j]="(+) ${NUMERO} de oros"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de oros"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                        2)

                                            JUGADOR2[i]=""

                                            ;;

                                        3)

                                            JUGADOR3[i]=""

                                            ;;

                                        4)

                                            JUGADOR4[i]=""

                                            ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

elif [[ "${OROS[j+1]}" == "(+) $((NUMERO + 1)) de oros" ]] || [[ "${OROS[j-1]}" == "(+) $((NUMERO - 1)) de oros" ]] ; then

                                OROS[j]="(+) ${NUMERO} de oros"

                                    echo "JUGADOR $TURNO PONE CARTA: ${NUMERO} de oros"

                                    CARTAS\_JUGADOR[i]=""

                                    case $TURNO in

                                    2)

                                        JUGADOR2[i]=""

                                        ;;

                                    3)

                                        JUGADOR3[i]=""

                                        ;;

                                    4)

                                        JUGADOR4[i]=""

                                        ;;

                                    esac

                                    TURNO=$((TURNO + 1))

                                    PUSO\_CARTA=true

*#clear*

                                    mostrar\_tablero

                                    break

                                fi

                            fi

                    done

                    ;;

                esac

            fi

        fi

done

if [ "$PUSO\_CARTA" == false ]; then

    echo "JUGADOR $TURNO PASA TURNO :("

    TURNO=$((TURNO + 1))

fi

}

La función selección secuencial se encarga de recorrer los arrays de las manos de la maquina de forma que comprueban carta por carta si es ponible sobre la mesa, para ello seleccionan una carta de la mano y recortan con “cut” palo y numero, según el palo de numero se dirige a uno d ellos palos sobre la mesa para buscar su igual en el tablero, una vez encontrado comprueba si el elemento i+1 e i-1 contienen un (+) en su inicio indicando que están colocadas, si esto es cierto la posición en i en el array del tablero se añade un (+) y se indica por un booleano que se ha colocado una carta para que no ponga más en su turno. Si este booleano sigue en false al final de el recorrido significa que no puede poner nada y se salta su turno indicándolo por pantalla.

Además, esta función contempla los casos especiales tanto del juego como de la baraja española, el primero que si el numero es un 5 este no necesita hacer comprobaciones con los contiguos de forma que se coloca automáticamente. Si la carta es un 10, tanto por nuestra implementación como por la baraja española su carta anterior no es un 9 por lo que en este caso debe buscar el numero 7 de su mismo palo como índice anterior.

function turno\_usuario {

    puede=false

    for ((i = 0; i < $NUMERO\_CARTAS\_JUGADOR1; i++)); do

        CARTA="${JUGADOR1[i]}"

            if [ -n "$CARTA" ]; then

                PALO=$(echo "$CARTA" | cut -d " " -f 3)

                NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

                if [ "$NUMERO" -eq 5 ]; then

                    puede=true

                    break

                else

                    case $PALO in

                        "bastos")

                            for ((j = 0; j < 10; j++)); do

                                if [[ "$CARTA" == "${BASTOS[j]}" ]]; then

                                    if [[ "$NUMERO" == "10" ]]; then

                                if  [[ "${BASTOS[j-1]}" == "(+) 7 de bastos" ]]; then

                                            puede=true

                                            break

                                        fi

elif [[ "${BASTOS[j+1]}" == "(+) $((NUMERO + 1)) de bastos" ]] || [[ "${BASTOS[j-1]}" == "(+) $((NUMERO - 1)) de bastos" ]] ; then

                                        puede=true

                                        break

                                    fi

                                fi

                            done

                            ;;

**//RESTO DE PALOS MISMO CODIGO**

if [ "$puede" = false ]; then

    echo "JUGADOR 1 PASA TURNO :("

    TURNO=2

    return

else

A continuación, en el caso de que sea el turno de nuestro usuario lo primero que realizamos es un recorrido para comprobar si tiene cartas posibles de manera que si nuestro booleano “puede” se encuentra en false este saltará de turno automáticamente esto lo realizamos con un switch para cada palo.

while true; do

    colocada=false

        read -p "Selecciona una carta para colocar (índice): " INDICE

        if [[ "$INDICE" =~ ^[0-9]+$ ]]; then

            if [ "$INDICE" -ge 1 ] && [ "$INDICE" -le ${NUMERO\_CARTAS\_JUGADOR1} ];then

                break

            else

                echo "Índice no válido. Debes seleccionar una carta entre 1 y ${NUMERO\_CARTAS\_JUGADOR1}"

            fi

        else

            echo "Índice no válido. Debes seleccionar una carta válida (número positivo)"

        fi

    done

fi

Una vez hemos comprobado que nuestro jugador puede colocar una carta le preguntamos por medio de read el índice de la carta que quiere seleccionar de su mano y pasamos a realizar comprobaciones primero comprobamos que la carta sea un número o números para que no nos puedan introducir valores o negativos o letras una vez hemos comprobado que nos han introducido un número válido verificamos si este se encuentra en el rango entre 1 y el número de cartas del jugador en caso de seguir siendo no válido se pedirá por teclado un nuevo índice indicando el error que hemos cometido.

if [ "$INDICE" -ge 1 ] && [ "$INDICE" -le "${NUMERO\_CARTAS\_JUGADOR1}" ]; then

    CARTA="${JUGADOR1[INDICE - 1]}"

    INDICE\_REAL=$((INDICE - 1))

    if [ -n "$CARTA" ]; then

        PALO=$(echo "$CARTA" | cut -d " " -f 3)

        NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

        if [ "$NUMERO" == "5" ]; then

            case $PALO in

                "bastos")

                    BASTOS[4]="(+) 5 de bastos"

                    echo "JUGADOR $TURNO PONE CARTA: 5 de bastos"

                    JUGADOR1[INDICE\_REAL]=""

                    ;;

                "copas")

                    COPAS[4]="(+) 5 de copas"

                    echo "JUGADOR $TURNO PONE CARTA: 5 de copas"

                    JUGADOR1[INDICE\_REAL]=""

                    ;;

                "espadas")

                    ESPADAS[4]="(+) 5 de espadas"

                    echo "JUGADOR $TURNO PONE CARTA: 5 de espadas"

                    JUGADOR1[INDICE\_REAL]=""

                    ;;

            esac

*#clear*

            mostrar\_tablero

        elif [[ "$NUMERO" == "10" ]]; then

            case $PALO in

                "bastos")

                    for ((j = 0; j < 10; j++)); do

                        if [[ "${BASTOS[j-1]}" == "(+) 7 de bastos" ]]; then

                            BASTOS[j]="(+) $NUMERO de bastos"

                            colocada=true

                            echo "JUGADOR $TURNO PONE CARTA: $NUMERO de bastos"

                            JUGADOR1[INDICE\_REAL]=""

                            break

                        fi

                    done

                    if [ "$colocada" == false ]; then

                        echo "No puedes colocar esa carta"

                        turno\_usuario

                    fi

                ;;

                "copas")

                    for ((j = 0; j < 10; j++)); do

                        if [[ "${COPAS[j-1]}" == "(+) 7 de copas" ]]; then

                            COPAS[j]="(+) $NUMERO de copas"

                            colocada=true

                            echo "JUGADOR $TURNO PONE CARTA: $NUMERO de copas"

                            JUGADOR1[INDICE\_REAL]=""

                            break

                        fi

                    done

                    if [ "$colocada" == false ]; then

                        echo "No puedes colocar esa carta"

                        turno\_usuario

                    fi

                ;;

                "espadas")

                    for ((j = 0; j < 10; j++)); do

                        if [[ "${ESPADAS[j-1]}" == "(+) 7 de espadas" ]]; then

                            ESPADAS[j]="(+) $NUMERO de espadas"

                            colocada=true

                            echo "JUGADOR $TURNO PONE CARTA: $NUMERO de espadas"

                            JUGADOR1[INDICE\_REAL]=""

                            break

                        fi

                    done

                    if [ "$colocada" == false ]; then

                        echo "No puedes colocar esa carta"

                        turno\_usuario

                    fi

                ;;

                "oros")

                    for ((j = 0; j < 10; j++)); do

                        if [[ "${OROS[j-1]}" == "(+) 7 de oros" ]]; then

                            OROS[j]="(+) $NUMERO de oros"

                            colocada=true

                            echo "JUGADOR $TURNO PONE CARTA: $NUMERO de oros"

                            JUGADOR1[INDICE\_REAL]=""

                            break

                        fi

                    done

                    if [ "$colocada" == false ]; then

                        echo "No puedes colocar esa carta"

                        turno\_usuario

                    fi

                    ;;

            esac

*#clear*

            mostrar\_tablero

        elif [[ ! "$NUMERO" == "5" && ! "$NUMERO" == "10" ]]; then

            case $PALO in

            "bastos")

                for ((j = 0; j < 10; j++)); do

if [[ "${BASTOS[j+1]}" == "(+) $((NUMERO+1)) de bastos" || "${BASTOS[j-1]}" == "(+) $((NUMERO-1)) de bastos" ]]; then

                        BASTOS[j]="(+) $NUMERO de bastos"

                        colocada=true

                        echo "JUGADOR $TURNO PONE CARTA: $NUMERO de bastos"

                        JUGADOR1[INDICE\_REAL]=""

                        break

                    fi

                done

                if [ "$colocada" == false ]; then

                    echo "No puedes colocar esa carta"

                    turno\_usuario

                fi

            ;;

//CODIGO PARA EL RESTO DE LOS PALOS IGUAL

else

    echo "Ya has colocado esa carta"

    turno\_usuario

fi

fi

Tras una nueva comprobación de que el índice es válido pasamos a verificar qué número hemos seleccionado y si es usable sobre el tablero para ello primero comprobamos si el número elegido es un 5 en ese caso se coloca automáticamente en la posición del array de la mesa correspondiente y se reemplazan nuestra mano con un vacío (“”) Si no es un 5 después comprobamos que sea 10 en cuyo caso comprobaremos que su índice inferior sea 7 en vez de 9 en caso de que cumpla las condiciones se coloca y pasa en nuestra mano a estar vacío ahora si no es un 5 o 10 pasamos a la comprobación genérica esta al igual que en la función de selección\_secuencial comprueba si el valor encontrado en el array del palo correspondiente tienen por encima o por debajo en su inicio el símbolo (+) pudiendo entonces colocar la carta.

En cualquiera de los casos anteriores si se coloca una carta es decir se cumplen las condiciones se cambia el booleano colocada por true de forma que si éste se mantiene en false significa que no podemos colocar esa carta.

Además, si el elemento en el array de nuestra mano que hemos seleccionado está vacío nos indica que ya hemos puesto esa carta. Tras haber colocado la carta correcta nuestro turno se incrementa y pasa el siguiente.

function comprobar\_vacios {

    VACIO=0

*# Comprueba el mazo del jugador 1*

    for ((k=0; k < $NUMERO\_CARTAS\_JUGADOR1; k++)); do

        if [ -n "${JUGADOR1[k]}" ]; then

            VACIO=0

            break

        else

            VACIO=1

        fi

    done

    if [ "$VACIO" == "1" ]; then

        return

    fi

*# Comprueba el mazo del jugador 2*

    for ((k=0; k < $NUMERO\_CARTAS\_JUGADOR2; k++)); do

        if [ -n "${JUGADOR2[k]}" ]; then

            VACIO=0

            break

        else

            VACIO=2

        fi

    done

    if [ "$VACIO" == "2" ]; then

        return

    fi

*# Comprueba el mazo del jugador 3 si hay al menos 3 jugadores*

    if [ "$JUGADORES" -ge 3 ]; then

        for ((k=0; k < $NUMERO\_CARTAS\_JUGADOR3; k++)); do

            if [ -n "${JUGADOR3[k]}" ]; then

                VACIO=0

                break

            else

                VACIO=3

            fi

        done

    fi

    if [ "$VACIO" == "3" ]; then

        return

    fi

*# Comprueba el mazo del jugador 4 si hay 4 jugadores*

    if [ "$JUGADORES" -eq 4 ]; then

        for ((k=0; k < $NUMERO\_CARTAS\_JUGADOR4; k++)); do

            if [ -n "${JUGADOR4[k]}" ]; then

                VACIO=0

                break

            else

                VACIO=4

            fi

        done

    fi

    if [ "$VACIO" == "4" ]; then

        return

    fi

}

Como dijimos anteriormente para comprobar la condición de victoria de un jugador cada vez que cambia de turno se llama esta función comprobar vacíos la cual se encarga dependiendo del número de jugadores que tengamos de ver si el array está vacío en caso de estarlo cambia el valor de la variable vacío al número de jugador en el cual lo ha detectado retornando automáticamente para finalizar la partida.

HORA\_FIN=$SECONDS

if [ "$VACIO" == "1" ]; then

    echo

    echo "====== HAS GANADO LA PARTIDA JUGADOR 1 ======"

fi

if [ "$VACIO" == "2" ]; then

    echo

    echo "====== HAS GANADO LA PARTIDA JUGADOR 2 ======"

fi

if [ "$VACIO" == "3" ]; then

    echo

    echo "====== HAS GANADO LA PARTIDA JUGADOR 3 ======"

fi

if [ "$VACIO" == "4" ]; then

    echo

    echo "====== HAS GANADO LA PARTIDA JUGADOR 4 ======"

fi

conteo\_puntos

escribir\_fichero

// DIBUJO DE GANADOR

echo "JUGADOR $VACIO HAS CONSEGUIDO $PUNTOS\_GANADOR puntos en $CONTADOR\_TURNOS turnos y $TIEMPO\_PARTIDA segundos!!!"

read

return

Una vez hemos determinado cuál ha sido el jugador que se ha quedado sin cartas la función de turnos pasa a esta parte donde dependiendo del valor de vacío indica quién ha sido el ganador y llama a otras dos funciones para contar puntos y escribir el registro en nuestro fichero.log.

Además, finaliza el contador de tiempo de la partida.

function conteo\_puntos {

    PUNTOS\_GANADOR=0

    for ((i = 0; i < $NUMERO\_CARTAS\_JUGADOR1; i++)); do

        CARTA="${JUGADOR1[i]}"

        NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

        PUNTOS\_GANADOR=$((PUNTOS\_GANADOR + NUMERO))

    done

    for ((i = 0; i < $NUMERO\_CARTAS\_JUGADOR2; i++)); do

        CARTA="${JUGADOR2[i]}"

        NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

        PUNTOS\_GANADOR=$((PUNTOS\_GANADOR + NUMERO))

    done

    if [ "$JUGADORES" -ge 3 ]; then

        for ((i = 0; i < $NUMERO\_CARTAS\_JUGADOR3; i++)); do

            CARTA="${JUGADOR3[i]}"

            NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

            PUNTOS\_GANADOR=$((PUNTOS\_GANADOR + NUMERO))

        done

    fi

    if [ "$JUGADORES" -eq 4 ]; then

        for ((i = 0; i < $NUMERO\_CARTAS\_JUGADOR4; i++)); do

            CARTA="${JUGADOR4[i]}"

            NUMERO=$(echo "$CARTA" | cut -d " " -f 1)

            PUNTOS\_GANADOR=$((PUNTOS\_GANADOR + NUMERO))

        done

    fi

}

La función conteo\_puntos se encarga de ir recorriendo con un bucle for los arrays de cartas de cada jugador contando el número de veces que ha encontrado una carta no nula de forma que estos serán los puntos que ha conseguido el ganador

function contar\_cartas {

    NUMERO\_CARTAS\_JUGADOR1=0

    NUMERO\_CARTAS\_JUGADOR2=0

    NUMERO\_CARTAS\_JUGADOR3=0

    NUMERO\_CARTAS\_JUGADOR4=0

    for ((i = 0; i < ${#JUGADOR1[@]}; i++)); do

        if [ -n "${JUGADOR1[i]}" ]; then

            NUMERO\_CARTAS\_JUGADOR1=$((NUMERO\_CARTAS\_JUGADOR1 + 1))

        fi

    done

    for ((i = 0; i < ${#JUGADOR2[@]}; i++)); do

        if [ -n "${JUGADOR2[i]}" ]; then

            NUMERO\_CARTAS\_JUGADOR2=$((NUMERO\_CARTAS\_JUGADOR2 + 1))

        fi

    done

    if [ "$JUGADORES" -ge 3 ]; then

        for ((i = 0; i < ${#JUGADOR3[@]}; i++)); do

            if [ -n "${JUGADOR3[i]}" ]; then

                NUMERO\_CARTAS\_JUGADOR3=$((NUMERO\_CARTAS\_JUGADOR3 + 1))

            fi

        done

    else

        NUMERO\_CARTAS\_JUGADOR3=\*

    fi

    if [ "$JUGADORES" -eq 4 ]; then

        for ((i = 0; i < ${#JUGADOR4[@]}; i++)); do

            if [ -n "${JUGADOR4[i]}" ]; then

                NUMERO\_CARTAS\_JUGADOR4=$((NUMERO\_CARTAS\_JUGADOR4 + 1))

            fi

        done

    else

        NUMERO\_CARTAS\_JUGADOR4=\*

    fi

}

Antes de escribir en nuestro fichero.log debemos contar por cada jugador cuantas cartas les ha quedado y almacenarlo en una variable en caso de que ese jugador no exista se le añade un\* a la variable;

function escribir\_fichero {

    FECHA=$(date +%d%m%Y)

    HORA=$(date +%H:%M:%S)

    TIEMPO\_PARTIDA=$SECONDS

    RONDAS=$((CONTADOR\_TURNOS / JUGADORES))

    GANADOR="$VACIO"

    PUNTOS="$PUNTOS\_GANADOR"

    contar\_cartas

echo "$FECHA|$HORA|$JUGADORES|$TIEMPO\_PARTIDA|$RONDAS|$GANADOR|$PUNTOS|$NUMERO\_CARTAS\_JUGADOR1-$NUMERO\_CARTAS\_JUGADOR2-$NUMERO\_CARTAS\_JUGADOR3-$NUMERO\_CARTAS\_JUGADOR4" >> "$LOG"

}

Una vez hemos recogido todos los datos como la fecha utilizando la función del sistema date cogiendo solo los campos día mes y año y la hora utilizando la misma función utilizando los campos horas minutos y segundos además del tiempo de partida con la función seconds iniciada al principio de esta las rondas cuyo valor se calcula con el contador colocado en cada 1 de los turnos dividido entre el número de jugadores el ganador el cual es el valor de vacío y los puntos que es la suma de las cartas restantes de cada array. Teniendo estos datos almacenados en variables realizamos un echo a nuestro fichero cuya dirección está almacenada en la variable $LOG utilizando el operador >> para no sobrescribir el archivo en caso de que tuviese más datos.

**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:

function estadisticas {

if [ -f "$LOG" ]; then

    if [ ! -s "$LOG" ]; then

        echo "El archivo de registro '$LOG' esta vacio juega una partida para ver las estadisticas"

        return

    fi

*# Número total de partidas jugadas según líneas*

*# grep -c . cuenta las líneas del archivo*

    TOTAL\_PARTIDAS=$(grep -c . "$LOG")

*# Media de los tiempos de todas las partidas jugadas*

    TOTAL\_TIEMPO=0

    MEDIA\_TIEMPO=0

    for ((i = 1; i <= $TOTAL\_PARTIDAS; i++)); do

*#sed -n no imprime lo procesado y {i}p imprime la línea i*

        TIEMPO\_PARTIDA=$(sed -n "${i}p" "$LOG" | cut -d "|" -f 4)

        TOTAL\_TIEMPO=$((TOTAL\_TIEMPO + TIEMPO\_PARTIDA))

    done

    MEDIA\_TIEMPO=$(($TOTAL\_TIEMPO / $TOTAL\_PARTIDAS))

*# Tiempo total invertido en todas las partidas*

    TIEMPO\_TOTAL=$TOTAL\_TIEMPO

*# Media de los puntos obtenidos por el ganador en todas las partidas*

    TOTAL\_PUNTOS=0

    MEDIA\_PUNTOS\_GANADOR=0

    for ((i = 1; i <= $TOTAL\_PARTIDAS; i++)); do

        PUNTOS\_GANADOR\_E=$(sed -n "${i}p" "$LOG" | cut -d "|" -f 7)

        TOTAL\_PUNTOS=$((TOTAL\_PUNTOS + PUNTOS\_GANADOR\_E))

    done

    MEDIA\_PUNTOS\_GANADOR=$(($TOTAL\_PUNTOS / $TOTAL\_PARTIDAS))

*# Porcentaje de partidas ganadas por cada jugador*

    VICTORIAS\_JUGADOR\_1=0

    VICTORIAS\_JUGADOR\_2=0

    VICTORIAS\_JUGADOR\_3=0

    VICTORIAS\_JUGADOR\_4=0

    PARTIDAS\_JUGADOR\_1=0

    PARTIDAS\_JUGADOR\_2=0

    PARTIDAS\_JUGADOR\_3=0

    PARTIDAS\_JUGADOR\_4=0

*# Contabiliza las victorias de cada jugador y las partidas jugadas*

    for ((i = 1; i <= $TOTAL\_PARTIDAS; i++)); do

        GANADOR=$(sed -n "${i}p" "$LOG" | cut -d "|" -f 6)

        case $GANADOR in

            1)

                VICTORIAS\_JUGADOR\_1=$((VICTORIAS\_JUGADOR\_1 + 1))

                ;;

            2)

                VICTORIAS\_JUGADOR\_2=$((VICTORIAS\_JUGADOR\_2 + 1))

                ;;

            3)

                VICTORIAS\_JUGADOR\_3=$((VICTORIAS\_JUGADOR\_3 + 1))

                ;;

            4)

                VICTORIAS\_JUGADOR\_4=$((VICTORIAS\_JUGADOR\_4 + 1))

                ;;

        esac

        PARTIDAS\_JUGADOR\_1=$((PARTIDAS\_JUGADOR\_1 + 1))

    done

        PORCENTAJE\_GANADAS\_1=$((VICTORIAS\_JUGADOR\_1 \* 100 / PARTIDAS\_JUGADOR\_1))

        PORCENTAJE\_GANADAS\_2=$((VICTORIAS\_JUGADOR\_2 \* 100 / PARTIDAS\_JUGADOR\_1))

        PORCENTAJE\_GANADAS\_3=$((VICTORIAS\_JUGADOR\_3 \* 100 / PARTIDAS\_JUGADOR\_1))

        PORCENTAJE\_GANADAS\_4=$((VICTORIAS\_JUGADOR\_4 \* 100 / PARTIDAS\_JUGADOR\_1))

    clear

    echo

    echo "      =========================="

    echo "              ESTADISTICAS      "

    echo "      =========================="

    echo

    echo "Número total de partidas jugadas: $TOTAL\_PARTIDAS"

    echo "Media de los tiempos de todas las partidas jugadas: $MEDIA\_TIEMPO segundos"

    echo "Tiempo total invertido en todas las partidas: $TIEMPO\_TOTAL segundos"

    echo "Media de los puntos obtenidos por el ganador en todas las partidas: $MEDIA\_PUNTOS\_GANADOR puntos"

    echo "Porcentaje de partidas ganadas del jugador 1: $PORCENTAJE\_GANADAS\_1%"

    echo "Porcentaje de partidas ganadas del jugador 2: $PORCENTAJE\_GANADAS\_2%"

    echo "Porcentaje de partidas ganadas del jugador 3: $PORCENTAJE\_GANADAS\_3%"

    echo "Porcentaje de partidas ganadas del jugador 4: $PORCENTAJE\_GANADAS\_4%"

else

    echo "El archivo de registro '$LOG' no existe o esta vacio"

fi

    echo

    read -p "Pulse una tecla para continuar..."

}

Para calcular la estadística nos hemos basado en un valor inicial el cual es el número de líneas de nuestro fichero este se obtiene a través de la función grep -c para buscar líneas seguido de . El cual se asegura de que esa línea no esté vacía en caso de que exista un enter en nuestro fichero. Para las siguientes consultas a nuestro fichero hemos utilizado la función sed (STREAM EDITOR) la cual nos permite recortar segmentos de este y poder pasarlos a variables tanto en el caso de tiempos como de puntos lo que hemos creado ha sido una variable para el total y otra para la media luego a través de un bucle for hasta el número de partidas hemos ido cogiendo con sed la línea y recortando con cut la columna que nos interese.

**sed -n “${i}p” “$LOG” -> -n hace que no tenga salida por pantalla y p indica que i representa lineas**

Para la siguiente sección de calcular el porcentaje de victorias a cada jugador es distinto ya que no en todas las partidas con el mismo número de jugadores Es por ello por lo que a la hora de recorrer nuestro fichero hemos buscado cuántas partidas a juegos cada 1 y cuántas ha ganado y a partir de ahí hemos calculado la estadística.

**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.

function clasificacion {

if [ -f "$LOG" ]; then

    if [ ! -s "$LOG" ]; then

        echo "El archivo de registro '$LOG' esta vacio juega una partida para ver las estadisticas"

        return

    fi

    PARTIDA\_MAS\_CORTA=""

    PARTIDA\_MAS\_LARGA=""

    PARTIDA\_MAS\_RONDAS=""

    PARTIDA\_MENOS\_RONDAS=""

    PARTIDA\_MAS\_PUNTOS=""

    PARTIDA\_MAS\_CARTAS=""

*# Partida más corta*

*# Ordena el archivo por el campo 4 (tiempo total) y coge la primera línea (la más corta) y la guarda en la variable*

    PARTIDA\_MAS\_CORTA=$(sort -t "|" -k 4 -n "$LOG" | head -n 1)

*# Partida más larga*

*# Ordena el archivo por el campo 4 (tiempo total) y coge la última línea (la más larga) y la guarda en la variable*

    PARTIDA\_MAS\_LARGA=$(sort -t "|" -k 4 -n -r "$LOG" | head -n 1)

*# Partida con más rondas*

*# Ordena el archivo por el campo 5 (rondas) y coge la última línea (la que más rondas tiene) y la guarda en la variable*

    PARTIDA\_MAS\_RONDAS=$(sort -t "|" -k 5 -n -r "$LOG" | head -n 1)

*# Partida con menos rondas*

*# Ordena el archivo por el campo 5 (rondas) y coge la primera línea (la que menos rondas tiene) y la guarda en la variable*

    PARTIDA\_MENOS\_RONDAS=$(sort -t "|" -k 5 -n "$LOG" | head -n 1)

*# Partida con más puntos*

*# Ordena el archivo por el campo 7 (puntos) y coge la última línea (la que más puntos tiene) y la guarda en la variable*

    PARTIDA\_MAS\_PUNTOS=$(sort -t "|" -k 7 -n -r "$LOG" | head -n 1)

*# Partida con más cartas*

    PARTIDA\_MAS\_CARTAS=$(cut -d "|" -f 8 "$LOG" | sort -t "-" -k 1 -n | tail -n 1)

    clear

"                                  Fecha|Hora|Jugadores|TiempoTotal|Rondas|Ganador|Puntos|CartasJugadores"

    echo "DATOS partida más corta:          $PARTIDA\_MAS\_CORTA "

    echo "DATOS partida más larga:          $PARTIDA\_MAS\_LARGA "

    echo "DATOS partida con más rondas:     $PARTIDA\_MAS\_RONDAS "

    echo "DATOS partida con menos rondas:   $PARTIDA\_MENOS\_RONDAS "

    echo "DATOS partida con más puntos:     $PARTIDA\_MAS\_PUNTOS "

    echo "DATOS partida con más cartas:     $PARTIDA\_MAS\_CARTAS "

    echo

    read -p "Pulse una tecla para continuar..."

fi

}

**sort -t "|" -k 7 -n -r "$LOG" ->**

* -t "|": Esta opción especifica que el carácter "|" se utiliza como el delimitador de campos en cada línea. Esto significa que sort tratará cada línea como si estuviera compuesta por varios campos separados por "|".
* -k 7: Esto indica que se debe ordenar en función del séptimo campo de cada línea.
* -n: Esta opción indica que se debe realizar una ordenación numérica en lugar de una ordenación alfabética.
* -r: Esta opción indica que la ordenación se debe realizar en orden descendente (de mayor a menor).

**Head/Tail -n 1** ->

* -n 1 especifica que se deben mostrar solo una línea. Si cambias el número después de -n, por ejemplo, -n 10, mostrará las últimas 10 líneas en lugar de una.

**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.

- Al contar con un algoritmo de burbuja para colocar las cartas en orden de la máquina este puede ser un poco ineficiente y en el ordenadores con recursos limitados puede llegar a tener un tiempo de ordenación algo alto y a lo mejor pudiese haber sido más óptimo utilizando otro tipo de algoritmos de ordenación.

**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.