Funciones Generales

```
Función centrado(cadena)
    cadena_cent <- centrar cadena en 260 caracteres
    imprimir cadena_cent
Fin de la función
```

```
Función imprime_resultado()
    Si ganaste entonces
        Mientras apuesta no sea 0 hacer
            limpiar la pantalla
            disminuir apuesta en 1
            aumentar enpartida en 1
            centrado("Deposito: " + dinero + " En partida: " + enpartida + " Apuesta: " + apuesta)
            esperar 0.05 segundos
        Fin del bucle
    Si perdiste entonces
        apuesta <- apuesta * -1
        Mientras apuesta no sea 0 hacer
            limpiar la pantalla
            aumentar apuesta en 1
            disminuir enpartida en 1
            centrado("Deposito: " + dinero + " En partida: " + enpartida + " Apuesta: " + apuesta)
            esperar 0.05 segundos
        Fin del bucle
    Fin de la condición
    ganaste <- Falso
    perdiste <- Falso
Fin de la función
```

```
Función comprueba_moroso(tipo_apuesta, interfaz, mensaje)
    Si tipo_apuesta no es un entero entonces
        tipo_apuesta = esdigito(tipo_apuesta, interfaz, mensaje)

    Mientras (apuesta + tipo_apuesta) sea mayor que enpartida hacer
        Limpiar la pantalla
        Mostrar ruleta
        Mostrar "Lo siento pero tienes que introducir un número de monedas menor que el que tienes en
partida."
        Mostrar mensaje
        Solicitar al usuario y guardar en tipo_apuesta
        tipo_apuesta = esdigito(tipo_apuesta, interfaz, mensaje)
```

Devolver tipo_apuesta

Función esdigito(comprobante, interfaz, mensaje)
Mientras comprobante no exista o comprobante no sea un dígito hacer
Limpiar la pantalla
Mostrar interfaz
Solicitar al usuario para continuar con "\nLo siento, esta opción solamente contempla números.\nVuelve a intentarlo, para ello pulsa Enter."
Mostrar mensaje
Solicitar al usuario y guardar en comprobante

Devolver comprobante convertido a entero

Funciones Blackjack

Función manos_inic(mano, carta1, carta2)
imprimir línea en blanco
Para cada línea en carta_rever hacer
centrado(mano[carta1][linea] + " " + mano[carta2][linea])
Fin del bucle
Fin de la función

Función manos_hit(mano)
Para cada línea en carta_rever hacer
imprimir línea en blanco
Para cada carta en mano hacer
imprimir(mano[carta][linea] + " ", sin salto de línea)
Fin del bucle
Fin del bucle
imprimir línea en blanco
Fin de la función

Función carta_rmdon()
carta_aleatoria <- elección aleatoria de baraja_use
eliminar carta_aleatoria de baraja_use

```
    devolver carta_aleatoria
Fin de la función



Función suma_valores(valores_extraidos, sumador)
    sumador <- 0
    prim_vez_as <- Falso
    Para cada carta en valores_extraidos hacer
        indice <- índice de carta[0] en valores
        sumador <- sumador + valores_num[indice]
        Si carta[0] es "A" y sumador < 11 entonces
            prim_vez_as <- Verdadero
            sumador <- sumador + 10
        Fin de la condición
    Fin del bucle
    Para cada carta en valores_extraidos hacer
        Si carta[0] es "A" y sumador > 21 y prim_vez_as es Verdadero entonces
            prim_vez_as <- Falso
            sumador <- sumador - 10
        Fin de la condición
    Fin del bucle
    devolver sumador
Fin de la función



Función imprime_manos_hit(valores_extraidos, sumador, mano)
    carta_aleatoria <- carta_rmdon()
    new_carta <- crea_cartas(carta_aleatoria)
    agregar carta_aleatoria a valores_extraidos
    agregar new_carta a mano
    sumador <- suma_valores(valores_extraidos, sumador)
    limpiar la pantalla
    manos_inic(mano_c, 1, 0)
    imprimir mesa
    Si mano es mano_j_1 entonces
        imprimir "1º Mano"
    Si mano es mano_j_2 entonces
        imprimir "2º Mano"
    Fin de la condición
    manos_hit(mano)
    devolver sumador
Fin de la función
```

```
Función crea_cartas(carta_aleatoria)
    Si carta_aleatoria[0] es "10" entonces
        crear carta con formato específico para "10"
    Sino
        crear carta con formato estándar
    Fin de la condición
    Si carta_aleatoria[1] es "♥" o "♦" entonces
        Para cada línea en carta hacer
            añadir color rojo a la línea
        Fin del bucle
    Sino
        Para cada línea en carta hacer
            añadir color negro a la línea
        Fin del bucle
    Fin de la condición
    devolver carta
Fin de la función

Función croupier_hit(sumador, mano)
    sumador_c <- 0
    sumador_c <- suma_valores(valores_extraidos_c, sumador_c)
    Mientras sumador_c < 17 y sumador_c <= sumador hacer
        sumador_c <- suma_valores(valores_extraidos_c, sumador_c)
        Si sumador_c < 17 y sumador_c <= sumador entonces
            imprimir línea en blanco
            carta_aleatoria <- carta_rmdon()
            new_carta <- crea_cartas(carta_aleatoria)
            añadir carta_aleatoria a valores_extraidos_c
            añadir new_carta a mano_c
            limpiar la pantalla
            manos_hit(mano_c)
            imprimir mesa
            manos_hit(mano)
            esperar 1.5 segundos
        Fin de la condición
    Fin del bucle
    devolver sumador_c
Fin de la función

Función ganador_bj(sumador)
    Si sumador > sumador_c y sumador <= 21 o sumador_c > 21 y sumador <= 21 entonces
        centrado("YOU WIN")
        esperar entrada del usuario
        ganaste <- Verdadero
    Si sumador es igual a sumador_c entonces
        centrado("Empate")
```

```
        esperar entrada del usuario
    Sino
        centrado("YOU LOSE")
        esperar entrada del usuario
        perdiste <- Verdadero
    Fin de la condición
Fin de la función

Función black_jack(valores_extraidos, sumador)
    Si valores_extraidos[0][0] es "A" y valores_extraidos[1][0] está en valores[9:] o
valores_extraidos[1][0] es "A" y valores_extraidos[0][0] está en valores[9:] entonces
        blackjack <- Verdadero
        sumador <- 21
    Fin de la condición
    devolver sumador
Fin de la función
```

Funciones Ruleta

```
Función gira_rulet()
    Si nr_num[contador] es menor que 10 entonces
        ruleta_anim <- reemplazar nr[contador] con bola en ruleta
    Sino
        ruleta_anim <- reemplazar nr[contador] con bola y un espacio en ruleta
    Fin de la condición
    imprimir ruleta_anim
Fin de la función
```

Funciones Slots

```
Clase Slots
    Método __init__()
        Definir slot_one, slot_two y slot_three con los valores correspondientes
    Fin del método

    Método roll()
        Inicializar a, b, c con 1
        Mientras a no sea divisible por 6, generar un número aleatorio entre 70 y 110 para a
        Mientras b no sea divisible por 6, generar un número aleatorio entre 60 y 90 para b
        Mientras c no sea divisible por 6, generar un número aleatorio entre 70 y 100 para c
        Inicializar i con 1
        Para cada número en el rango de a hacer
            Limpiar la pantalla
            Si i es igual a 6 entonces
                Reiniciar i a 0
                Mover el primer elemento de slot_one, slot_two y slot_three al final
            Fin de la condición
```

```
            Componer las listas s1, s2, s3 para imprimir
            Incrementar i en 1
            Llamar a la función layout con s1, s2, s3
            Esperar 0.02 segundos
        Fin del bucle
        Reiniciar i a 1
        Para cada número en el rango de b hacer
            Limpiar la pantalla
            Si i es igual a 6 entonces
                Reiniciar i a 0
                Mover el primer elemento de slot_two y slot_three al final
            Fin de la condición
            Componer las listas s1, s2, s3 para imprimir
            Incrementar i en 1
            Llamar a la función layout con s1, s2, s3
            Esperar 0.02 segundos
        Fin del bucle
        Reiniciar i a 1
        Para cada número en el rango de c hacer
            Limpiar la pantalla
            Si i es igual a 6 entonces
                Reiniciar i a 0
                Mover el primer elemento de slot_three al final
            Fin de la condición
            Componer las listas s1, s2, s3 para imprimir
            Incrementar i en 1
            Llamar a la función layout con s1, s2, s3
            Esperar 0.02 segundos más t^2 / 90000 segundos
        Fin del bucle
    Fin del método
Fin de la clase


Función layout(s1, s2, s3)
    Imprimir la interfaz gráfica del juego
    Para cada elemento l, m, r en s1, s2, s3 hacer
        Imprimir l, m, r en la interfaz gráfica
    Fin del bucle
    Imprimir apuesta y enpartida en la interfaz gráfica
Fin de la función

Función welcome()
    Definir la lista de bienvenida
    Para cada elemento c en la lista de bienvenida hacer
        Imprimir c
        Esperar 0.1 segundos
    Fin del bucle
    Esperar 1 segundo
```

Fin de la función

Función premios_slots()
  Si el tercer elemento de slot_one es igual al segundo elemento de slot_two y el primer elemento de slot_three es igual al segundo elemento de slot_two entonces
    Calcular la nueva apuesta
    Ganaste <- Verdadero
  Fin de la condición
  Si el primer elemento de slot_one es igual al segundo elemento de slot_two y el tercer elemento de slot_three es igual al segundo elemento de slot_two entonces
    Calcular la nueva apuesta
    Ganaste <- Verdadero
  Fin de la condición
  Si el segundo elemento de slot_one es igual al segundo elemento de slot_two y el segundo elemento de slot_three es igual al segundo elemento de slot_two entonces
    Calcular la nueva apuesta
    Ganaste <- Verdadero
  Fin de la condición
  Si el primer elemento de slot_one es igual al primer elemento de slot_two y el primer elemento de slot_three es igual al primer elemento de slot_two entonces
    Calcular la nueva apuesta
    Ganaste <- Verdadero
  Fin de la condición
  Si el tercer elemento de slot_one es igual al tercer elemento de slot_two y el tercer elemento de slot_three es igual al tercer elemento de slot_two entonces
    Calcular la nueva apuesta
    Ganaste <- Verdadero
  Fin de la condición
  Si ganaste entonces
    Imprimir "Has ganado ", apuesta, " monedas."
    Esperar la entrada del usuario
  Fin de la condición
Fin de la función

Algoritmo Principal

Mientras opc_casino no sea igual a "4" hacer
  Limpiar la pantalla
  Mostrar casino_intr

  Solicitar al usuario "¿A qué juego quieres jugar?: " y guardar en opc_casino
  Si opc_casino es igual a "4" entonces
    Salir del programa

  Limpiar la pantalla

```
    Mostrar casino_intr

    Si opc_casino no es igual a "3" entonces
        Mostrar "¿Con cuanto dinero quieres entrar a jugar?"
        Mostrar "Ahora mismo tienes {deposito}"
        Solicitar al usuario y guardar en enpartida
        enpartida = esdigito(enpartida, casino_intr, "¿Con cuanto dinero quieres entrar a jugar?\nAhora mismo tienes {deposito}")

    Mientras enpartida sea mayor que deposito hacer
        Limpiar la pantalla
        Mostrar casino_intr
        Mostrar "¿Con cuanto dinero quieres entrar a jugar?"
        Mostrar "Ahora mismo tienes {deposito}"
        Solicitar al usuario y guardar en enpartida
        enpartida = esdigito(enpartida, casino_intr, "¿Con cuanto dinero quieres entrar a jugar?\nAhora mismo tienes {deposito}")

        Si enpartida es mayor que deposito entonces
            Mostrar "Lo siento, pero las monedas con las que ingreses a la partida tiene que ser menor que el que tienes en el depósito."
            Solicitar al usuario para continuar

    deposito = deposito - enpartida

    Si opc_casino es igual a "1" entonces
        Definir salir como Falso
        Definir finish como Falso
        Definir seguir como "s"

        Mientras salir sea Falso hacer
            Limpiar la pantalla

            Si finish es Verdadero entonces
                Mostrar mesa
                Solicitar al usuario "¿Quieres jugar otra partida? (s/n): " y guardar en seguir
                Limpiar la pantalla

            Si seguir es igual a "s" entonces
                Mostrar mesa
                Mostrar "\n"
                Solicitar al usuario "¿Cuánto quieres apostar?: " y guardar en apuesta
                apuesta = esdigito(apuesta, mesa, "¿Cuánto quieres apostar?: ")

                Mientras apuesta sea mayor que enpartida hacer
                    Mostrar "\nLo siento pero la apuesta tiene que ser menor que las monedas que tengas en partida"
                    Solicitar al usuario para continuar
                    Limpiar la pantalla
```

```
                Mostrar mesa
                Mostrar "\n"
                Solicitar al usuario "¿Cuánto quieres apostar?: " y guardar en apuesta
                apuesta = esdigito(apuesta, mesa, "¿Cuánto quieres apostar?: ")
Limpiar la pantalla

Definir hit como "s"
Definir blackjack como Falso
Definir finish como Falso
Definir mano_j como lista vacía
Definir mano_c como lista vacía
Definir sumador_c como 0
Definir valores_extraidos_c como lista vacía
Definir valores_extraidos_j como lista vacía
Definir baraja_use como copia de baraja

Si no se cumple la condición anterior entonces
   Definir salir como Verdadero
   Llamar a la función vuelta_al_deposito()

Mientras finish sea Falso hacer

   Añadir carta_rever a mano_c

   Para i en el rango de 4 hacer
      Definir carta_aleatoria como resultado de carta_rmdon()
      Definir new_carta como resultado de crea_cartas(carta_aleatoria)

      Si i es par entonces
         Añadir new_carta a mano_j
         Añadir carta_aleatoria a valores_extraidos_j
      Sino
         Añadir new_carta a mano_c
         Añadir carta_aleatoria a valores_extraidos_c

   Llamar a manos_inic(mano_c, 1, 0)
   Mostrar mesa
   Llamar a manos_inic(mano_j, 0, 1)

   Definir sumador_j como resultado de suma_valores(valores_extraidos_j, sumador_j)

   Definir pairs como resultado de comparar el primer elemento de valores_extraidos_j con el segundo

   Definir sumador_j como resultado de black_jack(valores_extraidos_j, sumador_j)

   Si blackjack es Verdadero entonces
      Definir win como Verdadero
      Definir apuesta como apuesta * 3/2
```

```
        Solicitar al usuario "BLACKJACK"
        Llamar a imprime_resultado()
        Definir finish como Verdadero

    Sino si pairs es Verdadero entonces

        Solicitar al usuario "¿Quieres dividir la mano? (s/n): " y guardar en dividir

        Si dividir es "s" y pairs es Verdadero entonces

            Definir mano_j_1 como lista con el primer elemento de mano_j
            Definir mano_j_2 como lista con el segundo elemento de mano_j
            Definir sumador_j_1 como 0
            Definir sumador_j_2 como 0
            Definir valores_extraidos_j_1 como lista con el primer elemento de valores_extraidos_j
            Definir valores_extraidos_j_2 como lista con el segundo elemento de valores_extraidos_j

            Definir sumador_j_1 como resultado de imprime_manos_hit(valores_extraidos_j_1, sumador_j_1,
mano_j_1)

            Definir sumador_j_1 como resultado de black_jack(valores_extraidos_j_1, sumador_j_1)
            Si blackjack es Verdadero entonces
                Solicitar al usuario "BLACKJACK"
                Definir blackjack como Falso

            Mientras hit sea "s" y sumador_j_1 sea menor que 21 hacer

                Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

                Si hit es "s" entonces
                    Definir sumador_j_1 como resultado de imprime_manos_hit(valores_extraidos_j_1,
sumador_j_1, mano_j_1)

            Solicitar al usuario "Pasamos a la segunda mano"
            Definir hit como "s"
            Definir sumador_j_2 como resultado de imprime_manos_hit(valores_extraidos_j_2, sumador_j_2,
mano_j_2)

            Definir sumador_j_2 como resultado de black_jack(valores_extraidos_j_2, sumador_j_2)
            Si blackjack es Verdadero entonces
                Solicitar al usuario "BLACKJACK"

            Mientras hit sea "s" y sumador_j_2 sea menor que 21 hacer

                Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

                Si hit es "s" entonces
                    Definir sumador_j_2 como resultado de imprime_manos_hit(valores_extraidos_j_2,
sumador_j_2, mano_j_2)
```

```
    Definir blackjack como Falso
Si dividir no es igual a "s" y sumador_j no es igual a 21 entonces

    Mientras hit sea igual a "s" hacer

        Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

        Si hit es igual a "s" entonces

            Definir sumador_j como resultado de imprime_manos_hit(valores_extraidos_j, sumador_j,
mano_j)

            Si sumador_j es mayor que 21 entonces
                Definir hit como "n"
                Definir finish como Verdadero
                Definir lose como Verdadero
                Llamar a la función centrado con argumento Fore.RED + "YOU LOSE" + Fore.RESET
                Solicitar al usuario para continuar
                Llamar a la función imprime_resultado()

Si finish es Falso entonces

    Definir croupier_play como resultado de negar (sumador_j_1 >= 21 y sumador_j_2 >= 21)

    Si croupier_play es Verdadero entonces

        Limpiar la pantalla
        Llamar a manos_inic(mano_c, 1, 0)
        Mostrar mesa
        Si pairs es Verdadero y dividir es igual a "s" entonces
            Llamar a manos_hit(mano_j_1)
            Llamar a manos_hit(mano_j_2)
        Sino
            Llamar a manos_hit(mano_j)
        Esperar 2 segundos

        Eliminar el primer elemento de mano_c

        Limpiar la pantalla
        Llamar a manos_inic(mano_c, 0, 1)
        Mostrar mesa
        Si pairs es Verdadero y dividir es igual a "s" entonces
            Llamar a manos_hit(mano_j_1)
            Llamar a manos_hit(mano_j_2)
        Sino
            Llamar a manos_hit(mano_j)
        Esperar 2 segundos
```

```
        Definir sumador_c como resultado de black_jack(valores_extraidos_c, sumador_c)
        Definir blackjack como Falso

        Si pairs es Verdadero y dividir es igual a "s" entonces

            Si sumador_c no es igual a 21 entonces

                Definir sumador_c como resultado de croupier_hit(sumador_j_1, mano_j_1)

        Sino

            Si sumador_c no es igual a 21 entonces

                Definir sumador_c como resultado de croupier_hit(sumador_j, mano_j)

    Sino
        Eliminar el tercer elemento de mano_c

Si pairs es Verdadero y dividir es igual a "s" entonces

    Limpiar la pantalla
    Llamar a manos_hit(mano_c)
    Mostrar mesa
    Mostrar "1º Mano"
    Llamar a manos_hit(mano_j_1)
    Esperar 1.5 segundos
    Llamar a black_jack(valores_extraidos_j_1, sumador_j_1)
    Definir memo_apuesta como apuesta
    Si blackjack es Verdadero entonces
        Definir apuesta como apuesta * 3/2
        Definir win como Verdadero
        Definir blackjack como Falso
        Solicitar al usuario "BLACKJACK"
    Sino
        Llamar a ganador_bj(sumador_j_1)

    Llamar a imprime_resultado()

    Limpiar la pantalla
    Llamar a manos_hit(mano_c)
    Mostrar mesa
    Mostrar "2º Mano"
    Llamar a manos_hit(mano_j_2)
    Esperar 1.5 segundos
    Llamar a black_jack(valores_extraidos_j_2, sumador_j_2)
    Definir apuesta como memo_apuesta
    Si blackjack es Verdadero entonces
```

```
        Definir win como Verdadero
        Definir apuesta como apuesta * 3/2
        Solicitar al usuario "BLACKJACK"
            Sino
                Llamar a ganador_bj(sumador_j_2)
                Llamar a imprime_resultado()

Sino
    Llamar a ganador_bj(sumador_j)
    Llamar a imprime_resultado()

Definir finish como Verdadero

Si opc_casino es igual a "2" entonces

    Definir finish como "s"

    Mientras finish sea igual a "s" hacer

        Definir dict_apuesta_num como diccionario vacío
        Definir dict_apuesta_docena como diccionario con "Primera docena" : Falso, "Segunda docena" :
Falso, "Tercera docena" : Falso
        Definir dict_apuesta_colum como diccionario con "Columna 1" : Falso, "Columna 2" : Falso,
"Columna 3" : Falso
        Definir dict_apuesta_color como diccionario con "Rojo" : Falso, "Negro" : Falso
        Definir dict_apuesta_par como diccionario con "Pares" : Falso, "Impares" : Falso
        Definir dict_apuesta_bajo como diccionario con "Alto" : Falso, "Bajo" : Falso
        Definir menu_rul como "0"
        Definir apuesta como 0
        Definir tiempo como 0.001
        Definir rul_opt como entero
        Definir tiempo_rulet como número aleatorio entre 300 y 400
        Definir contador como 0

        Mientras menu_rul no sea igual a "7" hacer
            Limpiar la pantalla
            Mostrar ruleta

            Definir menu_ruleta como "¿A qué quieres jugar?\n
            1. Número
            2. Docenas
            3. Columnas
            4. Color
            5. Par o impar
            6. Bajo o alto
            7. Comenzar partida"

            Mostrar menu_ruleta
```

```
Solicitar al usuario y guardar en menu_rul

Según el valor de menu_rul hacer

  Caso "1":
    Mientras rul_opt no sea igual a 777 hacer
      Limpiar la pantalla
      Mostrar ruleta
      Mostrar "Introduce un número: "
      Mostrar "Para salir introduce '777'"
      Solicitar al usuario y guardar en rul_opt
      rul_opt = esdigito(rul_opt, ruleta, "Introduce un número: \nPara salir introduce '777'")

      Si rul_opt es menor que 37 entonces
        Solicitar al usuario "Introduce la apuesta para el número {rul_opt}: " y guardar en
dict_apuesta_num[rul_opt]
        dict_apuesta_num[rul_opt] = comprueba_moroso(dict_apuesta_num[rul_opt], ruleta,
"Introduce la apuesta para el número {rul_opt}: ")

        Incrementar apuesta por dict_apuesta_num[rul_opt]

      Sino
        Si rul_opt no es igual a 777 entonces
          Mostrar "Esta opción no está contemplada."
          Solicitar al usuario para continuar

  Caso "2":
    Limpiar la pantalla
    Mostrar ruleta
    Mostrar "¿A qué docena quieres jugar?:\n
    1. Primera docena (1-12)
    2. Segunda docena (13-24)
    3. Tercera docena (25-36)\n"

    Solicitar al usuario y guardar en rul_opt
    Solicitar al usuario "\nIntroduce la apuesta para la docena: " y guardar en apuesta_docena
    apuesta_docena = comprueba_moroso(apuesta_docena, ruleta, "\nIntroduce la apuesta para la
docena: ")

    Según el valor de rul_opt hacer
      Caso "1":
        Definir dict_apuesta_docena["Primera docena"] como apuesta_docena
      Caso "2":
        Definir dict_apuesta_docena["Segunda docena"] como apuesta_docena
      Caso "3":
        Definir dict_apuesta_docena["Tercera docena"] como apuesta_docena

    Incrementar apuesta por apuesta_docena
```

```
Caso "3":
    Limpiar la pantalla
    Mostrar ruleta
    Mostrar "¿A qué columna quieres jugar?:\n
    1. Primer columna
    2. Segunda columna
    3. Tercera columna \n"

    Solicitar al usuario y guardar en rul_opt
    Solicitar al usuario "\nIntroduce la apuesta para la columna: " y guardar en
apuesta_columna
    apuesta_columna = comprueba_moroso(apuesta_columna, ruleta, "\nIntroduce
la apuesta para la columna: ")

    Según el valor de rul_opt hacer
        Caso "1":
            Definir dict_apuesta_colum["Columna 1"] como apuesta_columna
        Caso "2":
            Definir dict_apuesta_colum["Columna 2"] como apuesta_columna
        Caso "3":
            Definir dict_apuesta_colum["Columna 3"] como apuesta_columna

    Incrementar apuesta por apuesta_columna

Caso "4":
    Limpiar la pantalla
    Mostrar ruleta
    Mostrar "¿A qué color quieres jugar?:\n
    1. Negro
    2. Rojo \n"

    Solicitar al usuario y guardar en rul_opt
    Solicitar al usuario "\nIntroduce la apuesta para el color: " y guardar en
apuesta_color
    apuesta_color = comprueba_moroso(apuesta_color, ruleta, "Introduce la apuesta
para el color: ")

    Según el valor de rul_opt hacer
        Caso "1":
            Definir dict_apuesta_color["Negro"] como apuesta_color
        Caso "2":
            Definir dict_apuesta_color["Rojo"] como apuesta_color

    Incrementar apuesta por apuesta_color

Caso "5":
    Limpiar la pantalla
    Mostrar ruleta
```

Mostrar "¿A qué quieres jugar, pares o impares?:\n
1. Pares
2. Impares \n"

Solicitar al usuario y guardar en rul_opt
Solicitar al usuario "\nIntroduce la apuesta para pares / impares: " y guardar en apuesta_par
apuesta_par = comprueba_moroso(apuesta_par, ruleta, "Introduce la apuesta para pares / impares: ")

Según el valor de rul_opt hacer
Caso "1":
Definir dict_apuesta_par["Pares"] como apuesta_par
Caso "2":
Definir dict_apuesta_par["Impares"] como apuesta_par

Incrementar apuesta por apuesta_par

Caso "6":
Limpiar la pantalla
Mostrar ruleta
Mostrar "¿A qué quieres jugar, alto o bajo?:\n
1. Alto
2. Bajo \n"

Solicitar al usuario y guardar en rul_opt
Solicitar al usuario "\nIntroduce la apuesta para alto / bajo: " y guardar en apuesta_bajo
apuesta_bajo = comprueba_moroso(apuesta_bajo, ruleta, "Introduce la apuesta para alto / bajo: ")

Según el valor de rul_opt hacer
Caso "1":
Definir dict_apuesta_bajo["Alto"] como apuesta_bajo
Caso "2":
Definir dict_apuesta_bajo["Bajo"] como apuesta_bajo

Incrementar apuesta por apuesta_bajo

Definir apuesta como apuesta multiplicada por -1

Para i en el rango de tiempo_rulet hacer

Incrementar contador por 1

Si contador es mayor que 36 entonces
Definir contador como 0

```
    Esperar tiempo segundos

    Si i es menor que tiempo_rulet / 1.5 entonces
        Continuar al siguiente ciclo

    Sino si i es mayor que tiempo_rulet / 1.5 y i es menor que (tiempo_rulet * 94) /
100 entonces
        Definir tiempo como tiempo elevado a la potencia de (99/100)

    Sino
        Definir tiempo como tiempo elevado a la potencia de (19/20)

    Limpiar la pantalla
    Llamar a la función gira_rulet()

Mostrar "Ha salido el número ", Back.WHITE + nr[contador] + Back.RESET

Definir num_aleatorio como nr_num[contador]

Para cada par numero, dinero en dict_apuesta_num hacer
    Si num_aleatorio es igual a numero entonces
        Incrementar apuesta por (dinero * 36)

Si num_aleatorio es menor que 13 y dict_apuesta_docena["Primera docena"] no es
Falso entonces
    Incrementar apuesta por (apuesta_docena * 3)
Sino si num_aleatorio es menor que 25 y dict_apuesta_docena["Segunda docena"]
no es Falso entonces
    Incrementar apuesta por (apuesta_docena * 3)
Sino si num_aleatorio es menor que 37 y dict_apuesta_docena["Tercera docena"]
no es Falso entonces
    Incrementar apuesta por (apuesta_docena * 3)

Si num_aleatorio está en columna1 y dict_apuesta_colum["Columna 1"] no es
Falso entonces
    Incrementar apuesta por (apuesta_columna * 3)
Si num_aleatorio está en columna2 y dict_apuesta_colum["Columna 2"] no es
Falso entonces
    Incrementar apuesta por (apuesta_columna * 3)
Si num_aleatorio está en columna3 y dict_apuesta_colum["Columna 3"] no es
Falso entonces
    Incrementar apuesta por (apuesta_columna * 3)

Si num_aleatorio está en rojo y dict_apuesta_color["Rojo"] no es Falso entonces
    Incrementar apuesta por (apuesta_color * 2)
Sino si dict_apuesta_color["Negro"] no es Falso entonces
    Incrementar apuesta por (apuesta_color * 2)
```

Si num_aleatorio es par y dict_apuesta_par["Pares"] no es Falso entonces
    Incrementar apuesta por (apuesta_par * 2)
Sino si dict_apuesta_par["Impares"] no es Falso entonces
    Incrementar apuesta por (apuesta_par * 2)

Si num_aleatorio es menor que 19 y dict_apuesta_bajo["Bajo"] no es Falso entonces
    Incrementar apuesta por (apuesta_bajo * 2)
Sino si dict_apuesta_bajo["Alto"] es Verdadero entonces
    Incrementar apuesta por (apuesta_bajo * 2)

Si apuesta es mayor que 0 entonces
    Mostrar "Tus beneficios son ", Fore.GREEN + str(apuesta) + Fore.RESET
    Solicitar al usuario para continuar
    Definir win como Verdadero
Sino
    Mostrar "Tus beneficios son ", Fore.RED + str(apuesta) + Fore.RESET
    Solicitar al usuario para continuar
    Definir lose como Verdadero
    Si apuesta es menor que 0 entonces
        Definir apuesta como apuesta multiplicada por -1

Llamar a la función imprime_resultado()

Solicitar al usuario "¿Quieres jugar otra partida? (S/N): " y guardar en finish

Si finish no es igual a "s" entonces
    Llamar a la función vuelta_al_deposito()

Si opc_casino es igual a "3" entonces

    Crear una nueva instancia de Slots y guardar en machine

    Llamar a la función welcome()
    Limpiar la pantalla
    Mostrar "\n\n\n"
    Llamar a la función centrado con argumento "Introduce cuánto dinero quieres tener en partida:"
    Solicitar al usuario y guardar en enpartida
    enpartida = esdigito(enpartida, "", "Introduce cuánto dinero quieres tener en partida:")
    Limpiar la pantalla
    Mostrar "\n\n\n"
    Llamar a la función centrado con argumento "Introduce la apuesta con la quieres jugar: "
    Mostrar "\n"
    Llamar a la función centrado con argumento "Recuerda que en este modelo de juego la apuesta es invariable"

```
    Llamar a la función centrado con argumento "En el caso de que quieras modificar la apuesta tendrás que salir "
    Llamar a la función centrado con argumento "al menú principal e introducirla de nuevo."
    Mostrar "\n"
    Solicitar al usuario y guardar en apuesta
    apuesta = esdigito(apuesta, "Introduce la apuesta con la quieres jugar: \nRecuerda que en este modelo de juego la apuesta es invariable\nEn el caso de que quieras modificar la apuesta tendrás que salir", "")
    Restar enpartida de deposito y guardar en deposito
    Guardar apuesta en memo_apuesta
    Definir finish como "s"

    Mientras finish sea igual a "s" hacer

        Definir apuesta como memo_apuesta
        Restar apuesta de enpartida y guardar en enpartida

        Llamar al método roll de machine
        Llamar a la función premios_slots()
        Llamar a la función imprime_resultado()

        Mostrar "\n"

        Si enpartida es menor o igual a 0 entonces
            Mostrar "Lo siento te has quedado sin monedas."
            Solicitar al usuario para continuar

        Sino
            Solicitar al usuario "¿Quieres jugar otra vez? (S/N): " y guardar en finish

            Si finish no es igual a "s" entonces
                Llamar a la función vuelta_al_deposito()
```