Proyecto entrega 1 – DISEÑO

ANDRES FELIPE RUGE

JORGE ESTEBAN MARTINEZ

NICOLAS AGUILAR CHAPARRO

ANTON PATRIGNANI

OSCAR DANILO MARTINEZ BERNAL

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

BOGOTA D.C.

2023

Diseño de TADs

Estructuras del proyecto con operaciones principales:

1. Risk

- TAD Risk
    - Conjunto mínimo de datos
        - Jugadores, lista de jugadores, almacena todos los jugadores de partida
        - Cartas, lista de tarjetas, cartas con comodines, misiones y cartas con territorio y fichas
        - Continentes, lista de continentes, tiene los 6 continentes del juego.
        - Partida, verdadero cuando la partida se ha iniciado, falso cuando no se ha empezado una partida.
        - Ganador, verdadero cuando hay un ganador, falso cuando no hay ganador
        - TurnoActual, numero entero de 0 a 5 que indica cual es jugador en turno
        - Totalturnos, numero entero, indica el total de turnos que se han realizado
    - Comportamiento (operaciones) del objeto
        - LanzarDado() retorna un numero aleatorio de 1 a 6
        - asignarGanador() termina la partida cuando hay un ganador
        - crearContinente(), crea un continente dentro del tablero
        - CrearTarjetas(tipo, territorio, ficha, misión) crea una carta de juego, puede indicar valores nulos para crear un tipo específico de tarjeta.

- AgregarTropas(jugador), agrega tropas a un jugador dadas las cartas que posee.
- CrearJugador(nombre, qJugadores), crea un jugador con su respectivo nombre y con relación a la cantidad de jugadores ingresados crear el batallón inicial.
- IniciarPartida(), pone la bandera de partida en verdadero
- EstadoPartida(), retorna si la partida ya ha sido inicializada
- turnoJugado(), aumenta en 1 el contador de turnos para avanzar al siguiente jugador en turno
- EsTurnoJugador(), retorna true si es el turno del jugador pasado en parametros de la funcion
- MoverFichasJugado(), mueve una cantidad de fichas de un jugador a un territorio de un continente en especifico. Ocupacion territorio
- TerritoriosLibres(), muestra territorios desocupados

2. Jugador

- TAD Jugador
  - Conjunto mínimo de datos
    - Color, indica el color del jugador (verde, azul, rojo, amarillo, negro, gris)
    - Cartas, lista de cartas del jugador
    - Fichas, lista de fichas del jugador
    - NombreJugador, cadena de caracteres que indica el nombre del jugador en la partida.
  - Comportamiento (operaciones) del objeto
    - Getters y setters de los datos del jugador
    - GetColor()
    - SetColor()
    - AgregarCarta(), agrega carta al jugador
    - AgregarFicha(), se le asigna ficha al jugador
    - MoverFicha(), usar una de las fichas del jugador

3. Carta

- TAD Carta
  - Conjunto mínimo de datos
    - TipoCarta, cadena de caracteres, indica el tipo de carta que es (normal, comodín, misión)
    - Territorio, cadena de caracteres territorio que muestra (cualquiera de los 42 disponibles), nulo si es una carta misión o comodín
    - Ficha, cadena de caracteres, indica el tipo de ficha que muestra (infantería, caballería, artillería), nulo si es una carta misión o comodín
    - Misión, cadena de caracteres, indica la misión inscrita en la carta
  - Comportamiento (operaciones) del objeto
    - //metodos de los getters y setters

4. Continente

- TAD Continente
    - Conjunto mínimo de datos
        - Territorios, lista de territorios, almacena los territorios que se encuentran en determinado continente
        - Nombre del continente, cadena de cárteres que tiene el nombre del continente.
    - Comportamiento (operaciones) del objeto
        - ocuparTerritorio(ficha, nTerritorio) recibe una Ficha de un jugador y la guarda en el territorio número nTerritorio.
        - AddTerritorio(nombre), crea un territorio dentro de un continente con relación al nombre que recibe.
        - MoverFicha(jugador, territorio1, territorio2), mueve una ficha del territorio1 al territorio vecino territorio2 de un respectivo jugador.
        - ObtenerNombre(), retorna el nombre del continente

5. Territorio

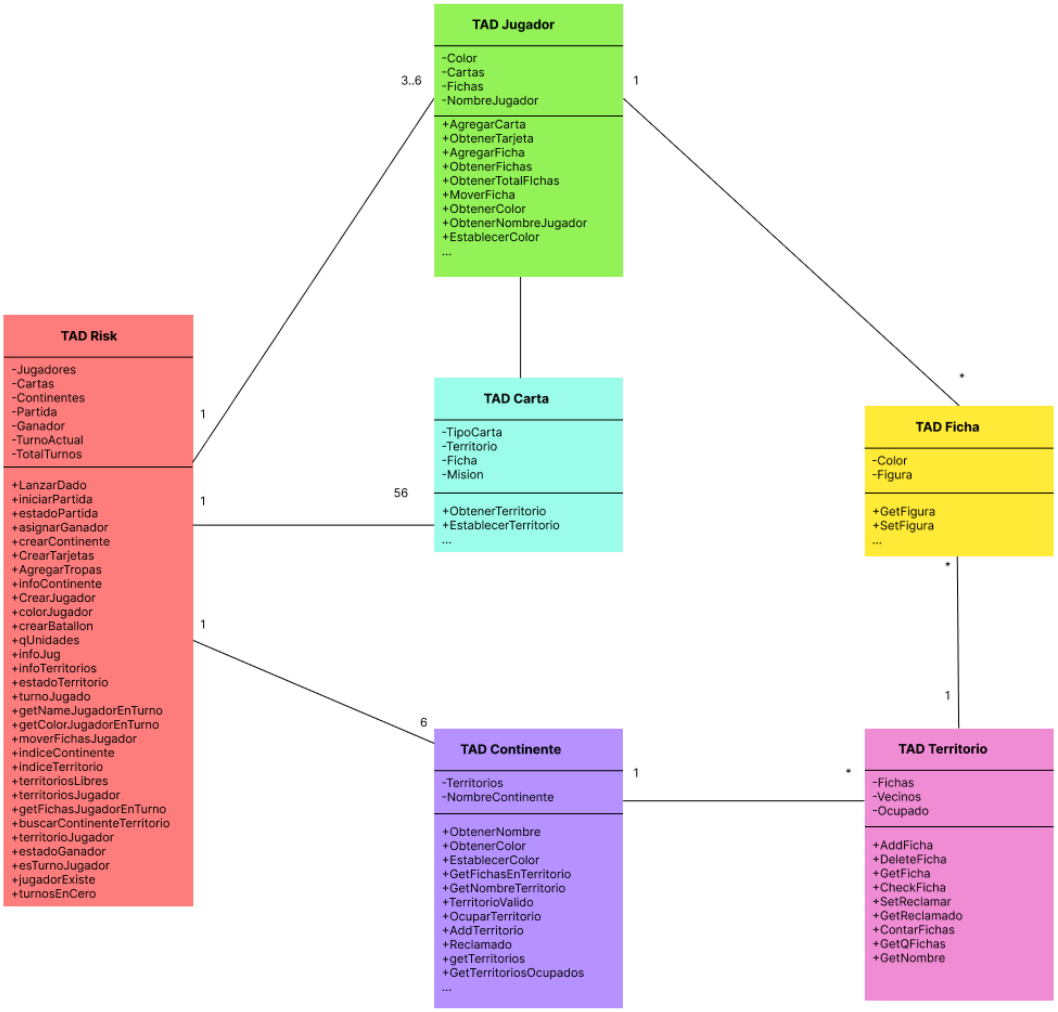
- TAD Territorio
    - Conjunto mínimo de datos
        - Fichas, lista de fichas que se encuentran dentro del territorio
        - Vecinos, cadena de caracteres que tiene el nombre de los países vecinos incluyendo los que se conectan con una línea recta.
        - Ocupado, cadena de caracteres que indica el nombre del jugador si el territorio ya ha sido reclamado por él, si no lo está, puede ser nulo.
    - Comportamiento (operaciones) del objeto
        - AddFicha(Ficha), agrega una Ficha de cualquier jugador al territorio.
        - deleteFicha(jugador), elimina una ficha de un jugador que se encuentre en el territorio.
        - GetFicha(jugador), busca y regresa la ficha de un jugador dentro del territorio, si no existe, regresa nulo.
        - ChekFicha(jugador), indica si existe o no una ficha de un jugador dentro del territorio.
        - SetReclamado(jugador), asígna el nombre del jugador que reclamó un territorio.
        - ContarFichas(jugador), cuenta la cantidad de fichas de un jugador dentro de un territorio.
        - GetQFichas(), retorna la cantidad de fichas totales dentro del territorio.

6. Ficha

- TAD Ficha
    - Conjunto mínimo de datos
        - Color, cadena de caracteres que indica el color asignado al jugador que pertenece.

    - Figura, cadena de caracteres que hace referencia al tipo de ficha que es (infantería, caballería, artillería).
- Comportamiento (operaciones) del objeto
    - GetColor(), retorna el color de la ficha (saber a cual jugador corresponde)
    - SetColor(color), recibe el color y este es asígnado a la ficha.

Diagrama de TADS:



Como podemos observar, la TAD principal, Risk, tiene relacion una a muchas con TADs Jugador, Carta, Continente, las cuales en si se relacionan con: Jugador – una a muchas con Carta y Ficha; Continente – una a muchas con Territorio, el cual tiene una conexión con Ficha. Hay que tener en cuenta los limites que pone el enunciado al momento de definicion de dichas TADs.

Procedimiento principal

El procedimiento principal consta de las siguientes suboperaciones (diagrama – mas abajo):

CrearContinente

Entradas: void

Salidas: void

Condiciones: crear nuevo Continente

IngresarComando

Entradas: void

Salidas: string cadena

Condiciones: no es null

Comandos del menu:

1) Inicializar con archivo

leerArchivo:

Entradas: string nombreArchivo

Salidas: void

Condiciones: nombreArchivo existe

2) Inicializar

InicializarJuego:

Entradas: Risk risk

Salidas: void

Condiciones: cantidad jugadores mayor o igual a 3 y hasta 6, continente exista, territorio exista en continente

Fortificar:

Entradas: Risk risk, bool init

Salidas: void

Condiciones: continente y territorio deben existir, fichas que el jugador desplaza deben ser menores a la cantidad total que tiene

3) Turno id_jugador

SeperarEspacio:

Entradas: string cadena, bool parametro

Salidas: string comando

Condiciones: subcadena.length > 0, subcadena dividida por space

4) Salir

5) Guardar nombre_archivo

CrearArchivo:

Entradas: string nombreArchivo

Salidas: void

Condiciones: creación de manera exitosa

6) Guardar_comprimido nombre_archivo

CrearArchivo:

Entradas: string nombreArchivo

Salidas: void

Condiciones: creación de manera exitosa

7) Help

8) Help comando

MostrarAyudaComando:

Entradas: string comando

Salidas: void

Condiciones: comando exista

Comandos y metodos

Risk

1) LanzarDado:

Descripcion - Lanzar un dado y retornar el número aleatorio de 1 a 6

Entradas - void

Salidas - int random

Condiciones - lanzar numero entero de 1 - 6

2) IniciarPartida:

Descripcion - pone la bandera de partida en verdadero

Entradas - void

Salidas - void

Condiciones - n/a

3) EstadoPartida:

Descripcion - retorna estado de la partida

Entradas - void

Salidas - bool

Condiciones - n/a

4) AsignarGanador:

Descripcion - asigna ganador y acaba partida

Entradas - void

Salidas - void

Condiciones - el jugador que gana tiene que existir

5) CrearContinente:

Descripcion - funcion para creación de continentes

Entradas - void

Salidas - void

Condiciones - utilizacion de continentes que existan el la vida real.

6) CrearTarjetas:

Descripcion - crear una carta de juego con los detalles proporcionados

Entradas - string tipo, string territorio, string ficha, string misión

Salidas - void

Condiciones - n/a

7) AgregarTropas:

Descripcion - agregar tropas a un jugador determinado al finalizar la ronda

Entradas - Jugador jugador

Salidas - n/a

Condiciones - el jugador debe existir

8) TurnoJugador:

Descripcion - aumenta la cantiad de turnos realizados, para poder avanzar al siguiente turno. Esta funcion realiza el avanzar en los jugadores mostrado el nombre del jugador.

Entradas - void

Salidas - void

Condiciones - n/a

9) InfoContinente:

Descripcion - retorna el nombre de un continente disponible

Entradas - ind indice

Salidas - string nombreContinente

Condiciones - n/a

10) InfoTerritorios:

Descripcion - guarda en una variable los territorios de un continente

Entradas - string nameContinente

Salidas - string retorno

Condiciones - ciclo de 6 iteracciones, el continente debe tener territorios

11) EstadoTerritorios:

Descripcion - retorna si el territorio el nombre del jugador quien ocupo el territorio o null si esta libre

Entradas - string nameContinente, string nameTerritorio

Salidas - bool estado

Condiciones - territorio y continente deben existir

12) CrearJugador:

Descripcion - recibo el nombre de un jugador y lo guardo en el vector de jugadores

Entradas - string nombre, int qJugadores

Salidas - void

Condiciones - n/a

13) ColorJugador:

Descripcion - retorna el color que se le debe asignar a un jugador

Entradas - void

Salidas - string color

Condiciones - n/a

14) QUnidades:

Descripcion - identifica las unidad ed de batallón inicial

Entradas - int qJugadores

Salidas - int unidad

Condiciones - n/a

15) InfoJug:

Descripcion - función de depuración

Entradas - void

Salidas - string

Condiciones - n/a

16) GetNameJugadorEnTurno:

Descripcion - obtener el nombre del jugador que tiene turno actual

Entradas - void

Salidas - string name

Condiciones - n/a

17) GetColorJugadorEnTurno:

Descripcion - obtener el color del jugador que tiene turno actual

Entradas - void

Salidas - string color

Condiciones - n/a

18) MoverFichasJugador:

Descripcion - mover ficha de un jugador a un territorio de un continente, cambiando el color de las tropas dependiendo del territorio

Entradas - int qFichas, string continente, string territorio

Salidas - bool

Condiciones - cantidad fichas mayor a 0, tiene que ser una cantidad de fichas que no exceda su cantidad de fichas actual del jugador actual.

19) IndiceContinente:

Descripcion - busca en el vector de continentes el indice de un continente

Entradas - string continente

Salidas - int indice

Condiciones - continente exista

20) IndiceTerritorio:

Descripcion - busca en el vector de territorio todos los indices

Entradas - int iContinente, string territorio

Salidas - int indice

Condiciones - indice de continente sea valido, territorio exista en el continente de entrada

21) TerritoriosJugador:

Descripcion - muestra los territorios del jugador actual en un continente

Entradas - string nameContinente

Salidas - string retorno

Condiciones - continente con el nombre nameContinente exista

22) CrearBatallon:

Descripcion – creacion del batallon de los jugadores

Entradas – int numero jugadores

Salidas - void

Condiciones - null

23) TerritoriosLibres:

Descripcion - indica si todos los territorios de todos los continentes han sido ocupados

Entradas - void

Salidas - bool

Condiciones – cantidad de ocupados mayor a 0

24) GetFichasJugadorEnTurno:

Descripcion - retorna la cantidad de fichas que tiene el jugador en turno

Entradas - void

Salidas - int

Condiciones - null

25) BuscarContinenteTerritorio:

Descripcion - busca y regresa el nombre del contiente del territorio que ingresó el usuario

Entradas – string territorio

Salidas – string continente

Condiciones – existan continentes y territorios

26) TerritorioJugador:

Descripcion - revisa que el territorio ingresado corresponda al jugador que se encuentra en turno

Entradas - string continente, string territorio

Salidas – bool status

Condiciones – existan continentes y territorios

27) EstadoGanador:

Descripcion – retorna el estado si se encontro el ganador

Entradas - void

Salidas – bool statusWinner

Condiciones - null

28) EsTurnoJugador:

Descripcion - retorna true, cuando el nombre ingresado es el mismo del jugador actual

Entradas – string nombreJug

Salidas – bool status

Condiciones – turnoActual > 0

30) TurnosEnCero:

Descripcion - pone el contador de turnos en cero para volver a empezar desde el jugador 0

Entradas - void

Salidas - void

Condiciones - null

2) Jugador

1) ObtenerColor:

Descripcion - obtencion del color de un jugador

Entradas - void

Salidas - string color

Condiciones - n/a

2) ObtenerNombreJugador:

Descripcion - obtencion del nombre de un jugador

Entradas - void

Salidas - string name

Condiciones - n/a

3) AgregarCarta:

Descripcion - agregar a las cartas que tiene el jugador una mas

Entradas - Carta carta

Salidas - void

Condiciones - n/a

4) AgregarFicha:

Descripcion - agregar una ficha mas al arreglo de fichas del jugador

Entradas - Ficha ficha

Salidas - void

Condiciones - n/a

5) ObtenerFichas:

Descripcion - retorna un vector de fichas

Entradas - void

Salidas - vector<Ficha> fichas

Condiciones - n/a

6) ObtenerTotalFichas:

Descripcion - obtener longitud del arreglo de fichas

Entradas - void

Salidas - int q

Condiciones - n/a

7) MoverFicha:

Descripcion - mover las fichas, retornando la primera

Entradas - void

Salidas - Ficha aux

Condiciones - existan fichas, fichas.length > 0.

3) Carta

1) getTerritorio:

Descripcion - obtener territorio de la carta

Entradas - void

Salidas - Territorio territorio

Condiciones - n/a

2) setTerritorio:

Descripcion - modificar territorios de la carta

Entradas - Territorio terr

Salidas - void

Condiciones - n/a

4) Continente

1) ObtenerNombre:

Descripcion - obtener nombre del continente

Entradas - void

Salidas - string name

Condiciones - n/a

2) setNombre:

Descripcion - modificar nombres de continentes

Entradas - string name

Salidas - void

Condiciones - n/a

3) OcuparTerritorio:

Descripcion - modificar estados relacionados con la ocupación de un territorio

Entradas - Ficha ficha, int nTerritorio, string color

Salidas - void

Condiciones - color sea igual a color usuario quien la atrapo, territorio con index nTerritorio debe existir

5) getNombreTerritorio:

Descripcion - obtener nombre del territorio

Entradas - int indice

Salidas - string name

Condiciones - territorio con indice exista

6) TerritorioValido:

Descripcion - buscar el nombre de un territorio en un continente y si existe, retorna true

Entradas - string territorio

Salidas - boolean status

Condiciones - n/a

7) AddTerritorio:

Descripcion - agrega territorio en un continente

Entradas - string territorio

Salidas - void

Condiciones - n/a

8) colorTerritorio:

Descripcion - obtener color de un territorio

Entradas - int terri

Salidas - string color

Condiciones - territorio con terri exista

9) GetFichasEnTerritorio:

Descripcion – obtener las fichas en un territorio determinado

Entradas – int indice

Salidas – int q

Condiciones – territorio[indice] exista

10) Reclamado:

Descripcion – obtener territorio reclamado

Entradas – int indice

Salidas - string

Condiciones - null

11) getTerritoriosOcupados:

Descripcion – retorna territorios con status ocupado

Entradas - void

Salidas – int index

Condiciones - null

5) Territorio:

1) AddFicha:

Descripcion - adicionar ficha

Entradas - Ficha ficha

Salidas - void

Condiciones - n/a

2) DeleteFicha:

Descripcion - eliminar ficha de la list de fichas de un jugador

Entradas - string jugador

Salidas - void

Condiciones - ficha exista en el territorio

3) GetFicha:

Descripcion - obtener la ficha final de un jugador

Entradas - string jugador

Salidas - Ficha ficha

Condiciones - jugador tiene que tener fichas en el territorio

4) CheckFicha:

Descripcion - verificar si existe la ficha del jugador

Entradas - string jugador

Salidas - bool status

Condiciones - ficha exista

5) SetReclamado:

Descripcion - cambiar estado del territorio como reclamado por un jugador en particular

Entradas - string jugador

Salidas - void

Condiciones - n/a

6) ContarFichas:

Descripcion - contar fichas de un jugador

Entradas - string jugador

Salidas - int q

Condiciones - n/a

7) GetQFichas:

Descripcion - obtener cantidad de fichas en un territorio

Entradas - void

Salidas - int cantidad

Condiciones - territorio exista

8) GetNombre:

Descripcion - obtener nombre del territorio

Entradas - void

Salidas - string nombre

Condiciones - n/a

9) AgregarVecino:

Descripcion - funcion para agregar un territorio vecino

Entradas - Territorio vecino

Salidas - void

Condiciones - territorio exista

6) Ficha

1) ObtenerColor:

Descripcion - obtener color de la ficha

Entradas - void

Salidas - string color

Condiciones - n/a

2) SetColor:

Descripcion - modificar color de la ficha

Entradas - string color

Salidas - void

Condiciones - n/a

3) ObtenerFigura:

Descripcion - obtener figura de la ficha

Entradas - void

Salidas - string figura

Condiciones - n/a

4) SetFigura:

Descripcion - modificar figura de la ficha

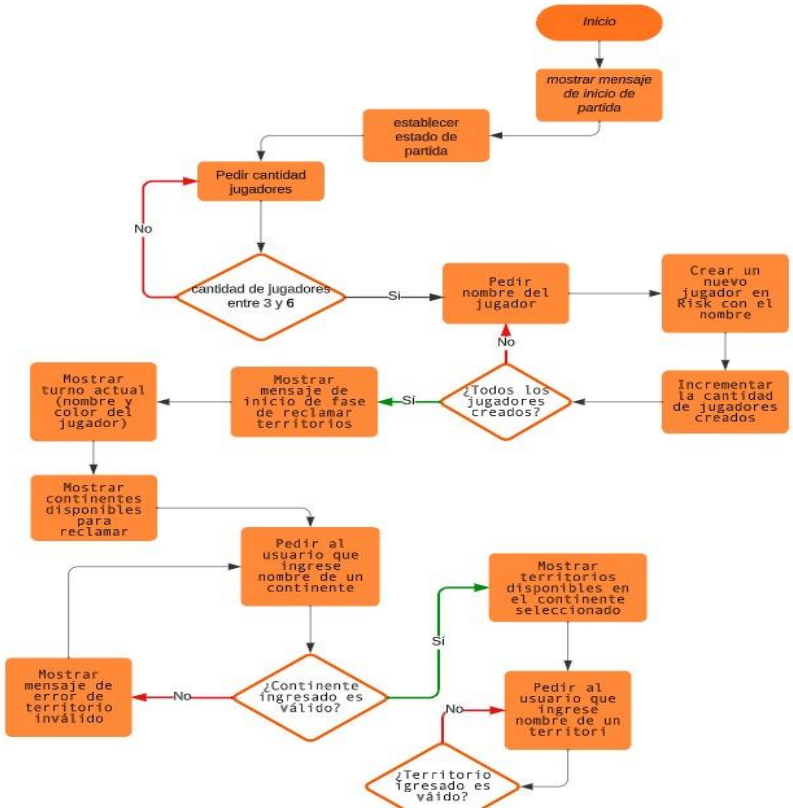
Entradas - string figura

Salidas - void

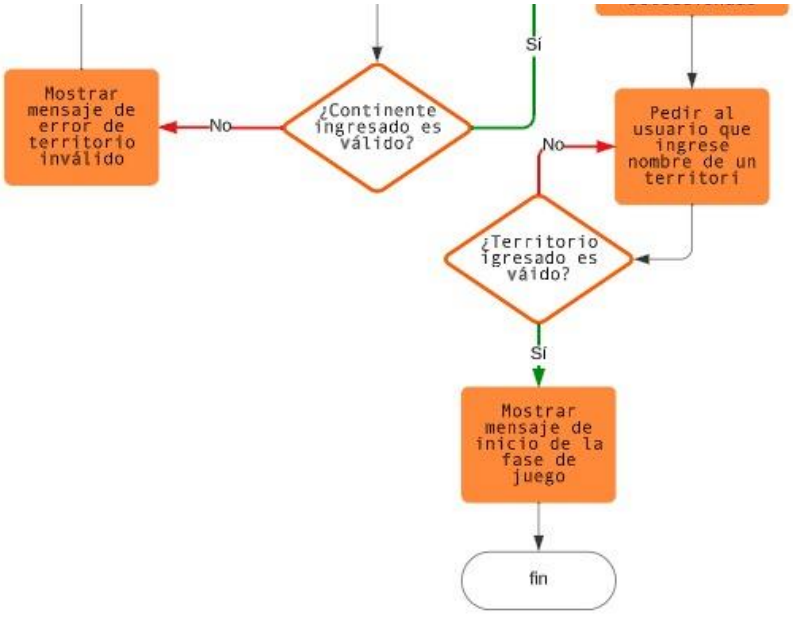
Condiciones - n/a

DIAGRAMAS DE FLUJOS

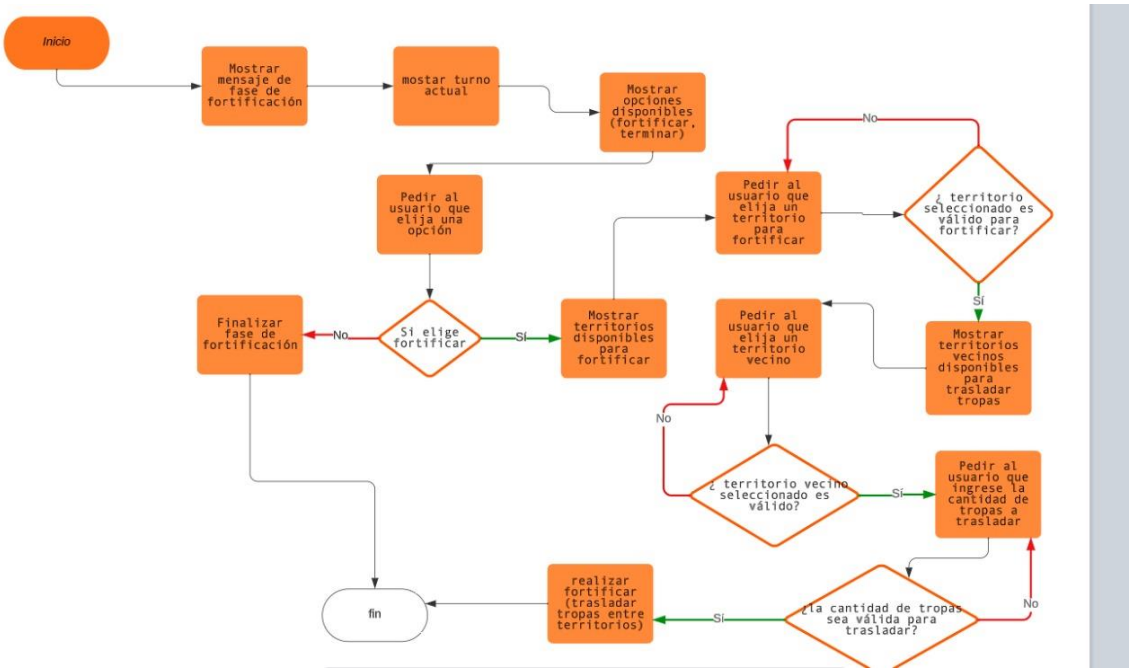
Inicializar – Parte Uno



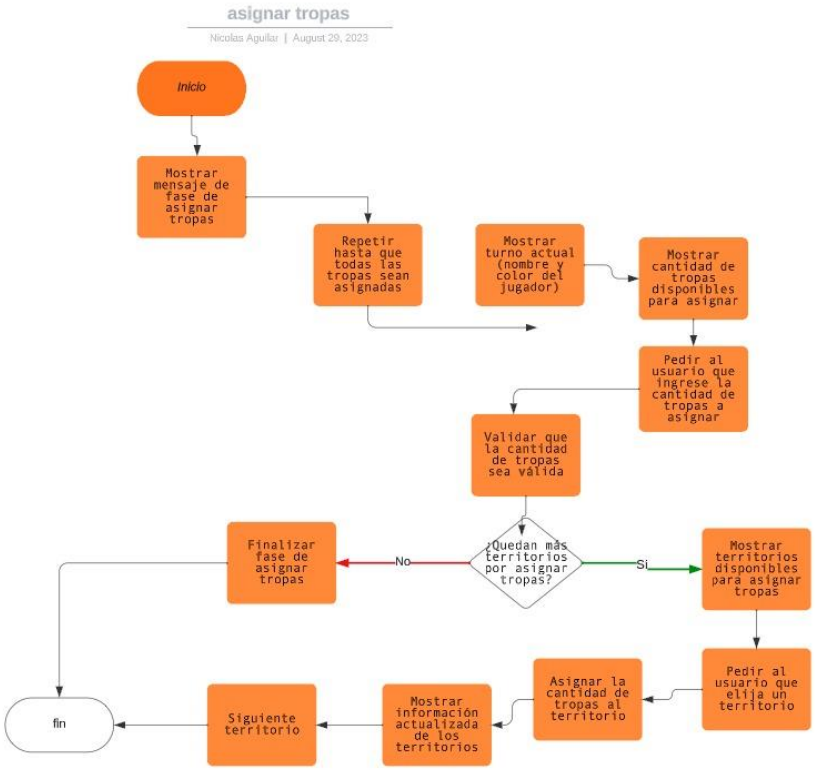
Inicializar – Parte Dos



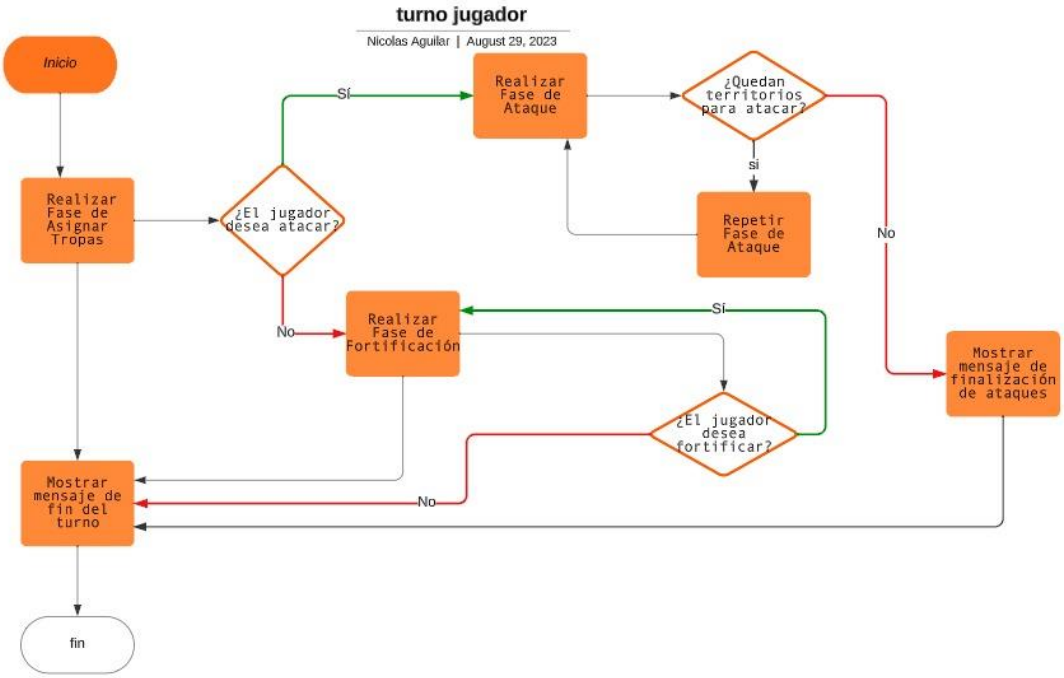
Fortificar



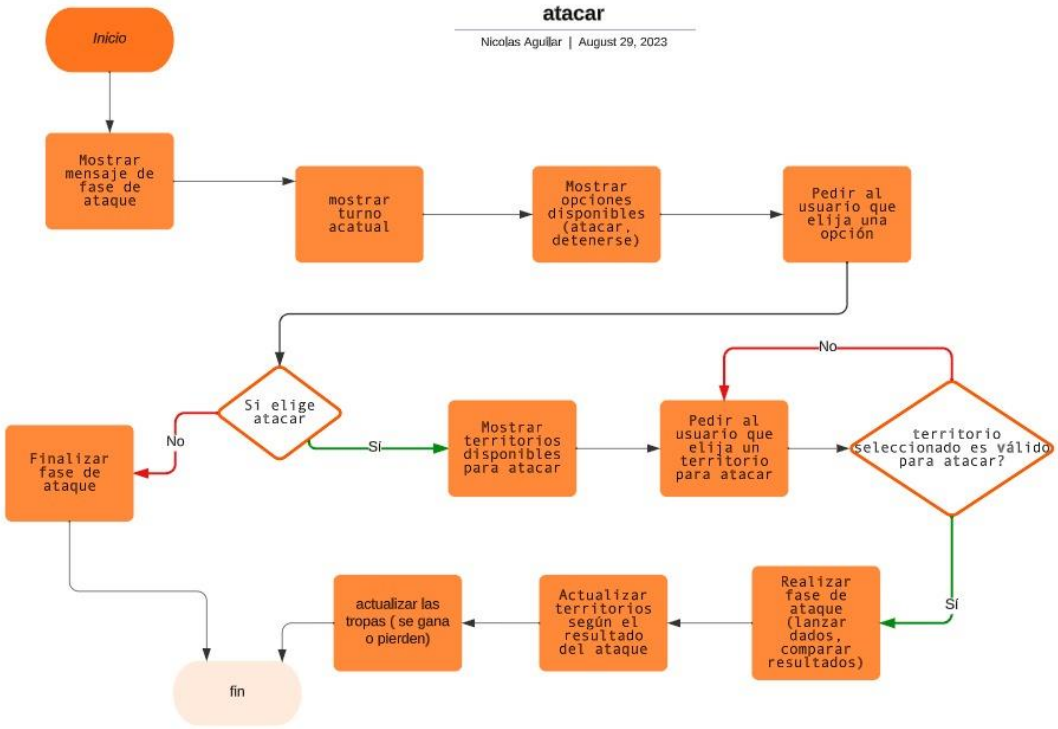
Asignar Tropas



Turno jugador



Atacar



PLAN DE PRUEBAS (INICIALIZAR)

Objetivo del plan de pruebas

Verificar el funcionamiento correcto de cada uno de los procesos esenciales en el programa, revisar posibles fallos o problemas algorítmicos y validar cada uno de estos procesos.

Identificación de requisitos

El programa debe cumplir con unos requisitos esenciales, dentro de los cuales están :

- Permitir jugar con una cantidad de jugadores entre 3 y 6.
- Permitir el acceso a cualquier comando que requiera el usuario según el menu de ayuda.
- Obtener la visualización de cada uno de los continentes y territorios del juego.
- Entre otros

Casos de prueba

- **Inicializar**

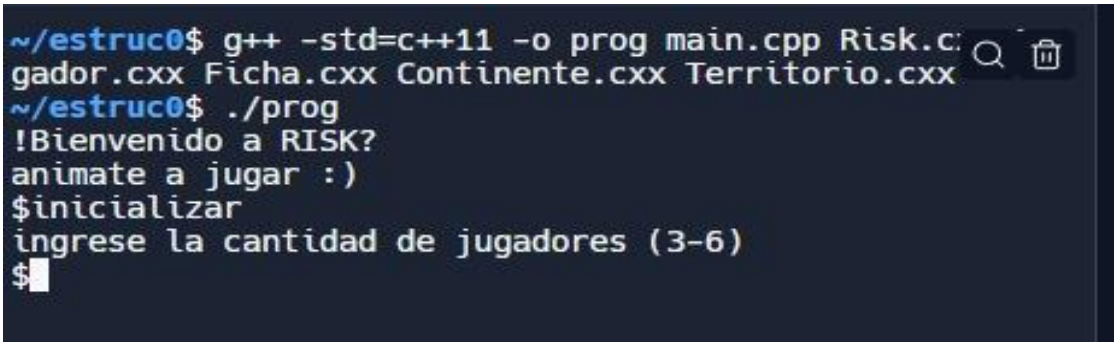
Resultado Esperado :

Al ejecutar el comando $inicializar en el programa, este deberá solicitarnos la cantidad de jugadores que habrá en la partida a crear, después de esto nos deberá permitir ingresar el nombre de cada jugador según la cantidad de jugadores y acto seguido se procederá con el reclamo de territorios de parte de cada jugador.

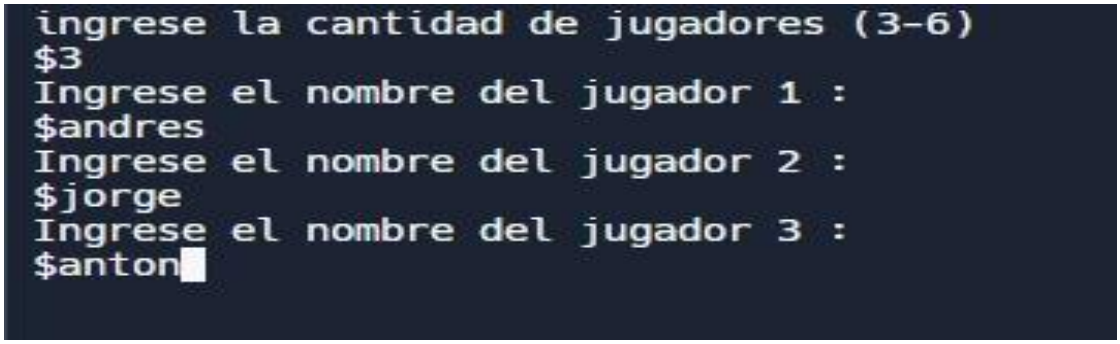
El proceso se hará de a turnos siguiendo el orden de ingreso de los nombres para cada turno, (por lo cual el primer jugador que ingreso su nombre será el primero en el orden del turno y así sucesivamente) en cada turno cada jugador podrá poner una ficha en el territorio de algún continente que este desee sin repetir territorios hasta que todo el mapa este completo.

Resultado Obtenido:

Como se puede evidenciar, el resultado del inicio fue igual al resultado esperado pidiendo observar cómo nos pide que ingresemos la cantidad de jugadores.



Acto seguido podemos ver cómo nos solicita los nombres de los jugadores según la cantidad de jugadores que se ingresó por lo cual este proceso también fue exitoso.



En la siguiente parte donde inicia el proceso de reclamar territorio, nuevamente el resultado obtenido es idéntico al resultado esperado, pudiendo observar cómo en un determinado turno el jugador de nombre “Jorge” puede elegir el continente disponible y acto seguido puede ver los territorios disponibles dentro de este continente.

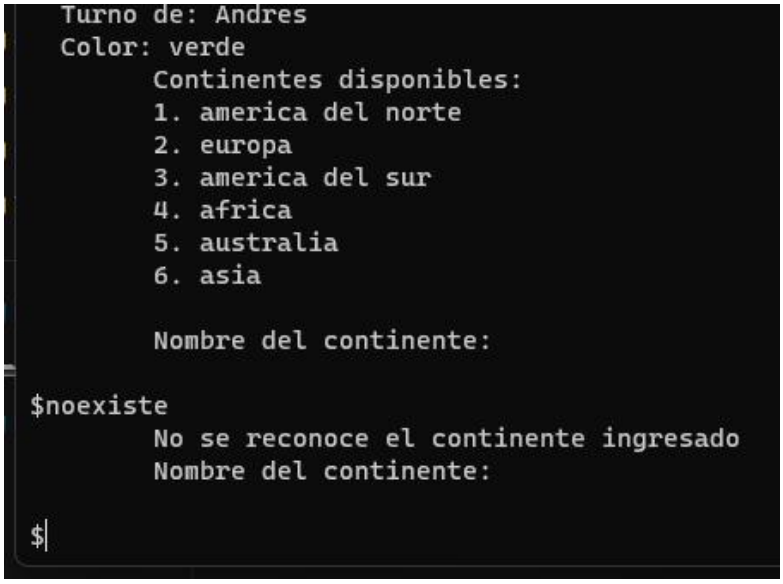
Posteriormente se evidencia que en las fichas del jugador Jorge se disminuyó una de sus fichas que ya fue colocada en ese territorio y después se pasa al turno del siguiente jugador que es “Anton” y así continuara el proceso hasta que se llene el mapa.

Casos de Error

La función de inicializar tiene implementada restricciones para casos no deseados como errores o malos ingresos de datos de parte del usuario.

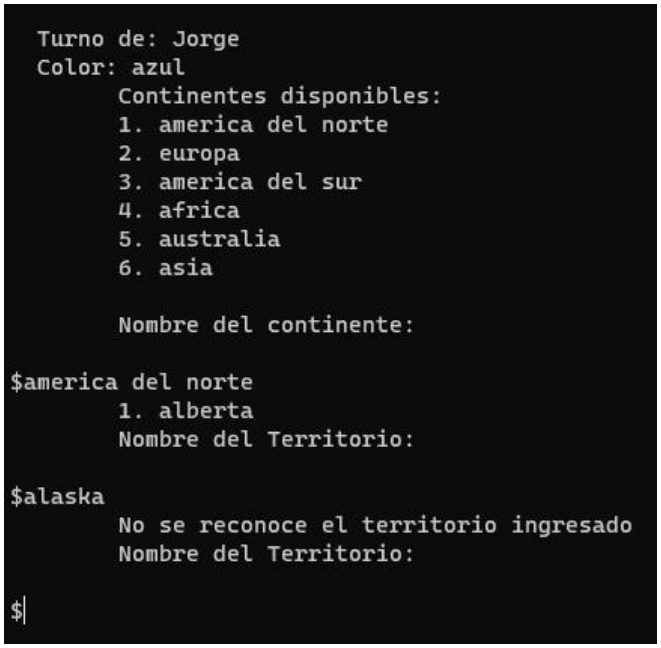
Algunas de estas restricciones son:

- Continente Invalido



Como se puede observar si el usuario digita un continente que no existe o algún otro tipo de carácter invalido, se le mostrara un mensaje indicando que el continente ingresado es invalido y le pedirá que ingrese nuevamente el continente deseado, por lo cual esta restricción se ha implementado de manera correcta y arroja el resultado esperado.

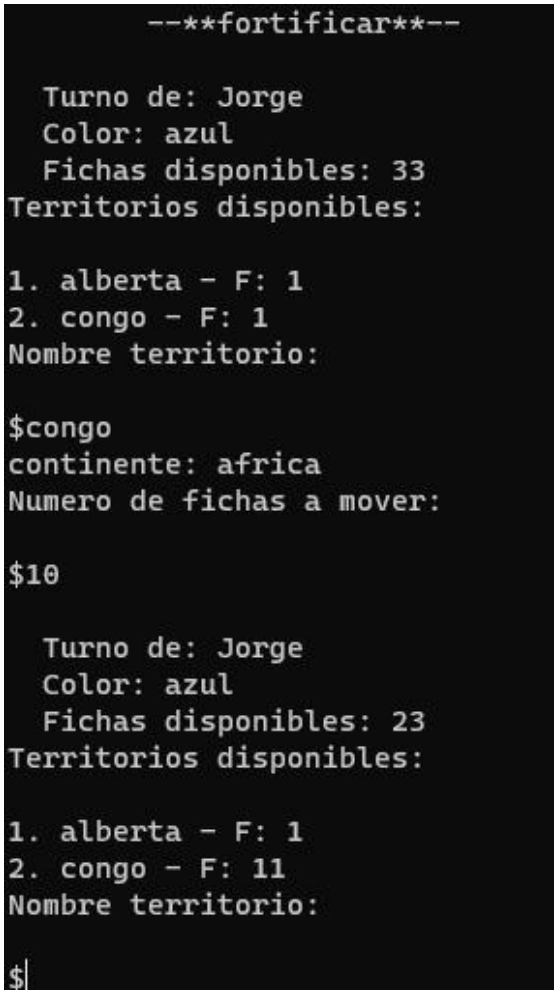
- Territorio Invalido



Al igual que pasa con el continente, si el usuario ingresa un territorio no existente, o que ya está ocupado por otro jugador o el mismo, o que es un carácter invalido, se le imprimirá un error diciendo que el territorio es invalido y se le solicitara que vuelva a digitar el territorio deseado.

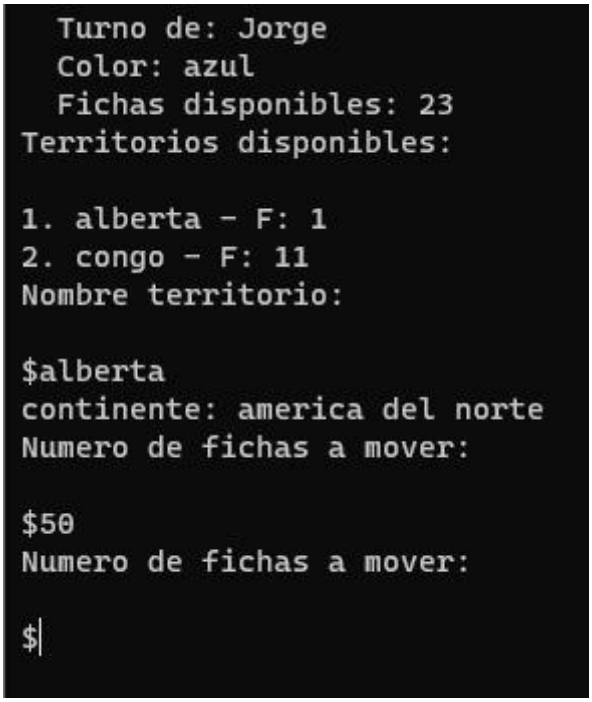
Final Prueba de Inicializar

Fortificar

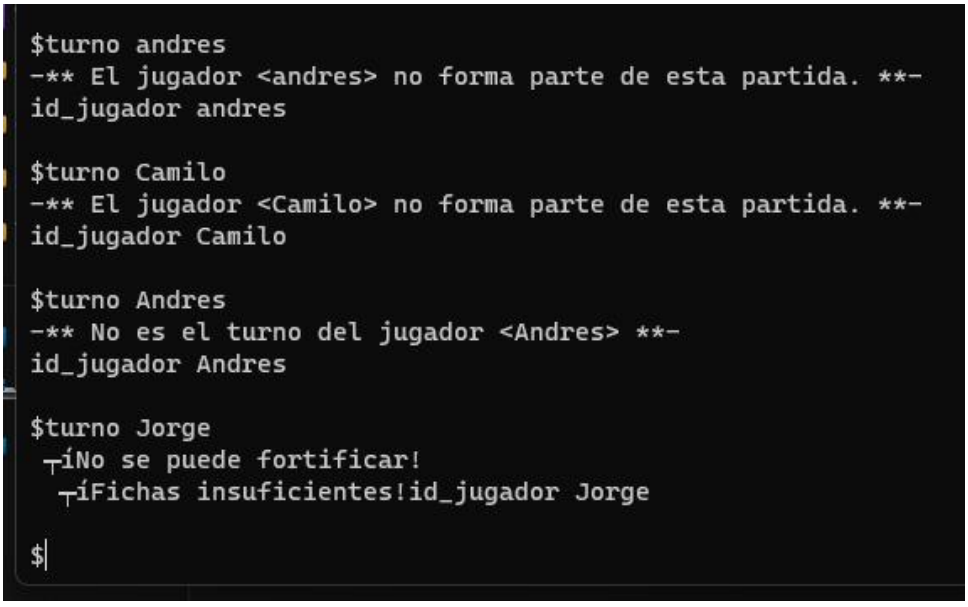


CASOS DE ERROR

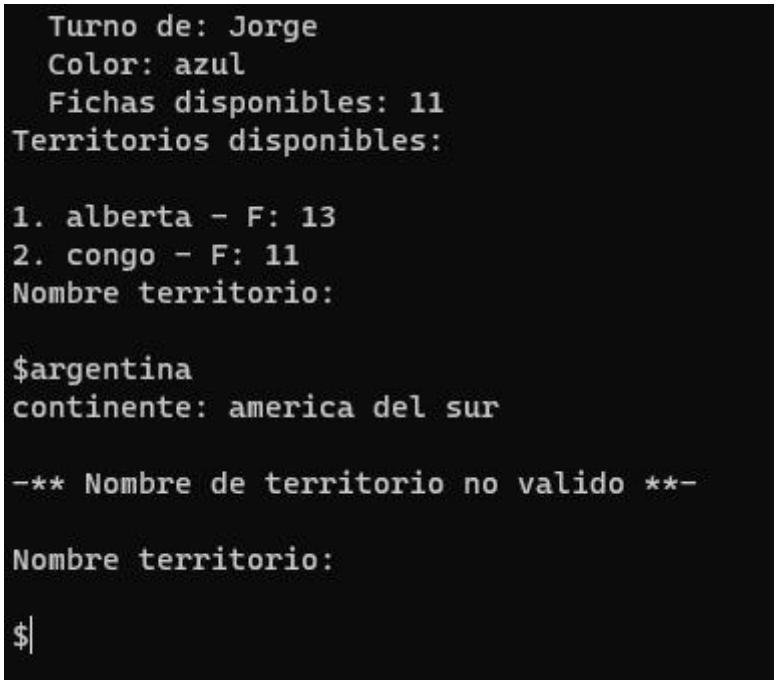
- Cuando el usuario solicita mover más fichas que tiene



- Turno invalido



- Fortificación de un territorio invalido



Turno

Un turno se identifica con una variable llamada turno actual, que no es más que un índice que va desde 0 a 5 en el TAD risk, saber cuál es el jugador en turno, el id del jugador ingresado debe corresponder al jugador en la posición “turno actual” dentro del vector de jugadores del TAD risk.

Para empezar un turno, debo usar la función