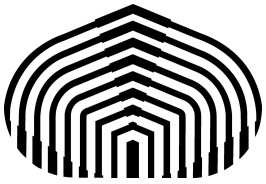

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la información
CI-2691- Laboratorio de algoritmos I

Proyecto - Reversi (Othello)

V. 1.0

El objetivo de este proyecto es ejercitar todas las actividades involucradas en el ciclo de vida de desarrollo de un programa. A diferencia de los laboratorios, en esta ocasión se presentará un problema para diseñar, implementar y probar una solución.

1. Planteamiento del problema

El juego de reversi, o también conocido como Othello por su publicación por Mattel, es un juego de mesa de estrategia que se realiza en un tablero 8x8. En cada una de las casillas se colocara una pieza ya sea en su lado negro o blanco, la cual determinará cuál de los jugadores posee esa casilla. El objetivo del juego es terminar con más casillas de tu color. Los jugadores toman turnos colocando fichas en el tablero, las cuales, cuando estas son jugadas, todas las fichas del otro color que se encuentren entre la ficha jugada y una ficha del mismo color, se convertirán en el color de la jugada realizada.

Se emplea un tablero de 8 filas por 8 columnas y 64 fichas idénticas, redondas, blancas por una cara y negras por la otra (u otros colores). Las casillas se denotan enumerando las columnas, comenzando por la esquina superior izquierda del tablero, con letras de la A a la H, e igual con las filas, pero con números del uno al ocho. A un jugador se le asigna un color y se dice que lleva las fichas de ese color, lo mismo para el adversario con el otro color.

Al inicio del juego se colocan cuatro fichas, tal como se ve en la Figura 1: dos fichas blancas en D4 y E5, y dos negras en E4 y D5.

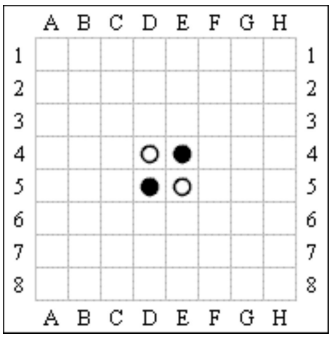

Figura 1

Empezando por quien lleva las fichas negras los jugadores deben hacer un movimiento por turno, a menos que no puedan hacer ninguno, pasando en ese caso el turno al jugador contrario. El movimiento consiste en colocar una ficha de forma que flanquee una o varias fichas del color contrario y voltear esas fichas para que pasen a mostrar el propio color.

Se voltean todas las fichas que se han flanqueado en ese turno al colocar la ficha del color contrario. Esas fichas, para que estén flanqueadas, deben formar una línea continua recta (diagonal u ortogonal) de fichas del mismo color entre dos fichas del color contrario (una de ellas la recién colocada y la otra ya presente). En el ejemplo de la figura 2 (izquierda) juegan las blancas donde indica la flecha y se puede ver qué fichas se voltean (derecha).

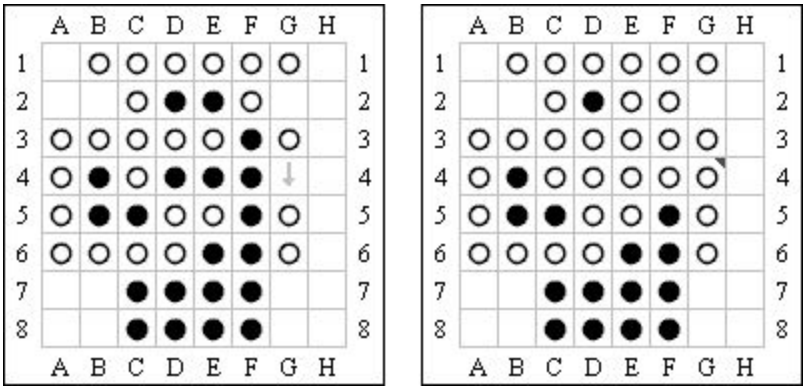

Figura 2

La partida finaliza cuando ningún jugador puede jugar (normalmente cuando el tablero está lleno de fichas) y gana quien en ese momento tenga sobre el tablero más fichas mostrando su color.

2. ¿Qué hacer primero?

Se trata de un proyecto de programación que involucra dos componentes básicas: una de cálculo, que contiene la parte de la lógica del juego y una de interfaz, con los elementos con los que se va a interactuar con la componente de cálculo. Este tipo de dicotomía se presenta con frecuencia en el desarrollo de programas.

En la componente de cálculo, lo primero que se debe hacer es tener una visión general de la solución. Aplicando análisis descendente se obtiene una primera versión:

```
do se desea jugar -->
        do quedan fichas  -->
                Obtener jugada;
                if jugada es válida -->
                        Realizar jugada;
                [] jugada no válida -->
                        Error
                fi
        od
        Resultado
od
```

Nótese que, aunque de manera muy general, esta primera versión ofrece una visión de la lógica del programa y permite detectar qué partes se deben desarrollar en detalle para obtener una solución más precisa. Este es precisamente el objetivo del análisis descendente: comenzar con problema grande y plantear su solución en función de subproblemas más sencillos.

A partir del análisis descendente, se pueden identificar los subproblemas más importantes: determinar si se desea jugar, si quedan fichas, si una jugada dada es válida, realizar una jugada, determinar las fichas a cambiar de color y mostrar el resultado final.

3. Estructuras de datos

Antes de proceder a la especificación de los subproblemas identificados en la parte 2, resulta conveniente decidir el tipo de datos que se usará para representar dos componentes principales del juego: el tablero y la jugada.

Un tablero puede considerarse como un arreglo de arreglos (o arreglo bidimensional) de enteros. En particular, cada casilla podrá tener uno de los tres valores siguientes:

- 0: para indicar que la casilla está vacía.
- 1: para indicar que la casilla está ocupada por una ficha del jugador 1.
- 2: para indicar que la casilla está ocupada por una ficha del jugador 2.

Adicional al tablero, hay que determinar cómo se va a representar una jugada. Una jugada está representada por el jugador en turno, la fila y la columna que el jugador seleccionó para poner una ficha. Para el jugador de turno se puede utilizar una variable de tipo booleano que tome uno de los dos valores siguientes:

- 0: para indicar que el jugador 1 está de turno.
- 1: para indicar que el jugador 2 está de turno.

Para los casos de la fila y la columna, también se usarán variables de tipo entero, que pueden tomar valores entre 1 y 8.

Una vez decidido el tipo de datos que conviene para representar un tablero y una jugada, se puede proceder a las especificación de los subproblemas identificados en la parte 2.

4. Subproblema de verificar si quedan fichas

Uno de los subproblemas que surgió del análisis descendente es el de determinar si en un momento dado es posible realizar una jugada. Para ello debe verificarse que aún no se han jugado las 64 fichas.

5. Subproblema de verificar si una jugada es válida.

Otro subproblema derivado del análisis descendente es el de determinar si la jugada suministrada por uno de los jugadores es válida.

Actividades a reportar :

- 5.1 Según la descripción del problema, ¿bajo qué condiciones una jugada es válida?
- 5.2 Proponga una especificación de una función EsValida, con la siguiente forma:

```
booleano func EsValida(entrada T: arreglo de arreglo de enteros,
                       Entrada fila, columna: entero)
      { Pre: …preEsValida…}
      { Post: …postEsValida…}
```

- 5.3 Escriba las instrucciones de la función EsValida.
- 5.4 Demuestre la correctitud de la función EsValida.

6. Subproblema para verificar si se consumen fichas

Tal vez el subproblema de verificar cuáles fichas se consumen (cambian de color) en una jugada pueda ser el más complejo. Para determinar que una o varias fichas son consumidas o cambiadas al otro color, en la última jugada tiene que haber sido rodeada por la ficha jugada y una presente anteriormente, ya sea de forma vertical, horizontal o diagonal.

Aplicando el análisis descendiente podemos identificar 3 subproblemas: consumo vertical, consumo horizontal, consumo diagonal.

Actividades a reportar:

- Proponga una especificacion para cada uno de los 3 subproblemas. Use los nombres consumoVertical, consumoHorizontal, consumoDiagonal.
- Escriba las instrucciones de cada uno de los subprogramas.
- Demuestre la correctitud de la función consumoVertical.
- Proponga una especificacion de la funcion consumo, con la siguiente forma

```
Booleano func consumo(entrada T: arreglo de arreglo de enteros,
                      entrada fil, col, turno: entero)
      { Pre: …preConsumo…}
      { Post: …postConsumo…}
```

- Escriba las instrucciones de la función consumo. Haga uso de las funciones ya definidas.

7. Una partida

Con los subproblemas hasta ahora definidos, se puede completar el conjunto de instrucciones necesarias para realizar una partida. Para llevar a cabo una jugada, se debe saber de quién es el turno. De esta forma, cada vez que se realiza una jugada válida, se cambia el jugador en turno.

Actividades a reportar :

- 7.1 Proponga una especificación de un procedimiento cambiarJugador con la siguiente forma:

```
proc cambiarJugador (entradasalida turno : entero)
      { Pre: …PreCambiarJugador…}
      { Post: …PostCambiarJugador…}
```

- 7.2 Escriba las instrucciones de la función cambiarJugador.

Por último, se debe especificar un procedimiento reflejarJugada que muestre la jugada del jugador de turno sobre el tablero.

Actividades a reportar:

- 7.3 Proponga una especificación de un procedimiento reflejarJugada con la siguiente forma:

```
proc reflejarJugada (entrada T : arreglo de arreglos de enteros;
entrada turno: entero)
      { Pre: …PreRealizarJugada…}
      { Post: …PostRealizarJugada…}
```

- 7.4 Escriba las instrucciones del procedimiento reflejarJugada.

Supóngase que se tiene una función obtenerJugada que captura del usuario la jugada que desea realizar.

7.5 Incorporando estos desarrollos, el programa resulta:

```
[ var T: arreglo de arreglo de arreglo de enteros;
 fil, col: entero;
 turno, fichas : entero;
{...PrePartida...}
do quedanFichas →
      obtenerJugada(fil, col);
      if esValida(T, fil, col) →
            consumo(T, fil, col, turno)
            reflejarJugada(fil, col, turno);
            cambiarJugador(turno)
      [] no esValida(T, fil, col) →
            Error
      Fi
od ;
Resultado
```

8. Resultado

Imprime el marcador final con el total de puntos o casillas conseguidas por cada uno. Para ello, calcularPuntos actualiza dos contadores, uno para cada jugador, cada vez que se realiza una jugada. calcularPuntos debe imprimir el resultado parcial cada vez que se ejecuta.

9. Jugar varias partidas

Ahora se puede escribir una secuencia de instrucciones que permita jugar varias partidas. Supóngase que se tiene una función OtraPartida, que captura de los jugadores si desea jugar otro partido. El programa quedaría así:

```
[ var T: arreglo de arreglo de enteros;
 fil, col: entero;
 turno, fichas : entero;]
{…PreReversi…}
turno := 0;
otro := verdad;
do otro →
      {…PrePartida…}
      do quedanFichas →
            obtenerJugada(fil, col);
            if esValida(T, fil, col) →
```

```
                consumo(T, fil, col, turno);
                reflejarJugada(fil, col, turno);
                cambiarJugador(turno);
            [] no esValida(T, fil, col) →
                Error
            Fi
      od ;
      Resultado
      {…PostPartida…}
      otro := otraPartida();
od;
{…PostReversi…}
```

Actividades a reportar:

- 9.1 Especifique el programa anterior. Si considera que debe agregar alguna instrucción para satisfacer PrePartida puede hacerlo, siempre y cuando lo justifique.
- 9.2 Nótese que hasta los momentos no se ha considerado el requerimiento de quién comienza cada partida. Determinar si en el programa anterior el requerimiento se cumple; en caso contrario, especifique y escriba el conjunto de instrucciones necesarias para incorporar este requerimiento.

10. Fin de la lógica. ¿Qué sigue?

Con las actividades anteriores se ha diseñado un programa que captura la parte lógica o de cálculo del problema del juego. Nótese que hasta el momento no se ha dado importancia a la manera en que los jugadores van a interactuar. Tan sólo se han introducido marginalmente un par de instrucciones, obtenerJugada y otraPartida, que capturan información de los jugadores.

La segunda parte en la elaboración del proyecto consiste en introducir los elementos de interfaz necesarios para que el programa pueda ser utilizado efectivamente.

Para facilitar la programación se limitarán las operaciones de entrada de información por teclado en modo texto. Para el despliegue de información se utilizarán dos modos de manera conjunta: modo texto y modo gráfico.

11. Elementos de la interfaz

Para solicitar los nombres de los jugadores sus nombres, pueden usarse instrucciones provistas por el lenguaje para tal fin y almacenar los nombres en dos variables nombreJugador1 y nombreJugador2 de tipo string. Luego, cada vez que le toque el turno al jugador 1, se puede desplegar un mensaje como:

Introduzca su jugada, NombreJugador1, por favor:

Análogamente se puede hacer para el jugador 2. También se podría dar mensaje en caso de que una jugada no sea válida.

Cada vez que se dé el resultado, final o parcial, se deben utilizar los nombres de los jugadores para presentar sus puntos. Además se puede preguntar si se desea jugar otra partida.

Actividad a reportar:

- 11.1 Indique en qué parte del programa colocaría las instrucciones de lectura de los nombres de los jugadores y el mensaje de jugada no válida.
- 11.2 Indique en qué parte del programa colocaría los mensajes que solicitan la jugada a un jugador. Diga si es necesario agregar instrucciones adicionales para controlar el despliegue del mensaje.

Para el caso del tablero, se utilizará una modificación de la máquina de trazados, que permite dibujar círculos rellenos con diferentes colores y borrar la máquina de trazados.

La nueva especificación del método dibujarCirculo es:

```
método dibujarCirculo(entrada x,y,r: entero; entrada c: color)
{Pre: r <=0 & |x| + r <= this.XMAX & |y| + r <= this.YMAX & c es un
color válido}
{Post: hayCirculo(this,x,y,r,c) }
```

donde color puede ser negro o blanco

Actividades a reportar:

- 11.3 Especifique y defina un procedimiento dibujarTablero que dibuje un tablero vacío para jugar a La Vieja.
- 11.4 Para dibujar una jugada particular es necesario determinar la coordenada del centro del círculo en función de la jugada actual. El color del círculo depende del jugador de turno. Especifique y defina un procedimiento que dibuje en el tablero una jugada. El encabezado del procedimiento debería ser:

```
proc dibujarJugada(entradasalida mt: Maquina de trazados;
 entrada T : arreglo de arreglos de enteros;
 entrada Fila, Columna, Turno: entero)
```

  El radio del círculo puede establecerse según como se considere más adecuada su visualización.

- 11.5 Incorporar las instrucciones dibujarTablero y dibujarJugada en el programa obtenido.

Por tratarse de un proyecto de programación de cierta complejidad, se recomienda que se haga la programación de forma incremental, de manera que se pueda probar la el funcionamiento de cada uno de los procedimientos y funciones.

En primer lugar se recomienda programar los elementos de la interfaz, en particular aquellos que permiten obtener una jugada y desplegarla en la pantalla. De esta manera se puede verificar que las instrucciones que calculan la posición en el tablero de la última jugada funcionan correctamente. Luego, agregar la función EsValida y verificar su funcionamiento, ejecutando tanto jugadas válidas como no válidas.

Hasta este punto el programa es capaz de ejecutar una sola partida. Una vez probada esta versión reducida del programa en Python, se puede proceder a agregar los elementos que permiten jugar varias partidas y calcular las estadísticas.

12. Documentación

El programa deberá estar documentado apropiadamente, siguiendo las normas de documentación del lenguaje Python. Y siguiendo las normas vistas en las clases del laboratorio.

13. Entregas

Deberá presentarse el esqueleto del programa en pseudocódigo y en Python, con las llamadas a los procedimientos desarrollados o no. También deberá estar desarrollado el subproblema para verificar si hay línea (al menos en pseudo lenguaje), así como las función esValida, con las especificaciones y pruebas requeridas.

Para la clase de la semana 10 se debe tener la interfaz gráfica y obtenerJugada, pudiendo verificar gráficamente la misma, así como dibujarTablero y dibujarJugada. Se revisará el esqueleto de programa con las nuevas incorporaciones. También debe entregarse todos los requerimientos solicitados de pseudocódigo, especificaciones y pruebas de correctitud.

También se deberá llevar una versión operativa del programa en Python que permita jugar a Reversi según las especificaciones indicadas en el planteamiento del problema. Es importante que el equipo trabaje de manera integrada. Durante la revisión se verificará el funcionamiento del programa y se hará un corto interrogatorio particular a ambos miembros del equipo.

Además, deberá entregar un informe que describa en detalle todo el proceso de diseño y desarrollo de la solución. El informe no se debe limitar a reportar las actividades indicadas en este enunciado; las actividades deben servir de guía para elaborar el informe. En este sentido, se quiere que se entregue un informe que contenga las siguientes secciones:

- Portada. Una hoja que contenga:
  Arriba a la izquierda el nombre de universidad, carrera y materia.
  Centrado: nombre del proyecto.

Abajo a la izquierda: nombre del profesor de laboratorio.
Abajo a la derecha: nombre y carnet de los integrantes del equipo

- Introducción:
  Breve descripción del problema atacado en el proyecto.
  Motivación.
  Objetivos planteados.
  Alcance de la solución.
  Contenido del informe.
- Diseño:
  Resultado del análisis descendente del problema.
  Especificación y desarrollo de los programas, procedimientos y funciones.
  Demostraciones de correctitud.
  Optimizaciones.
- Detalles de implementación.
  Tipos de datos.
  Estructuración del código.
  Cualquier comentario adicional referente a la traducción de los programas a Python.
- Estado actual.
  Operatividad del programa: decir si funciona perfectamente o no. En caso negativo, describir las anomalías.
  Manual de operación: nombre del archivo a ejecutar y modo de operar el programa.
- Conclusiones.
  Resultados obtenidos.
  Experiencias adquiridas.
  Dificultades presentadas.
  Recomendaciones.
- Bibliografía. Libros, revistas o cualquier recurso bibliográfico consultado. Una referencia bibliográfica debe incluir la siguiente información:
  Libro: autor(es), nombre, editorial y año.
  Artículo: autor(es), nombre, revista, volumen, número y año.
  Página Web: autor(es), nombre, url, fecha última visita.

  Los siguientes son ejemplos de referencias bibliográficas:
  - Libro:
    Blair, D.: Language and Representation in Information Retrieval. Elsevier Science Publisher, 1990.
  - Revista:
    Isaacson, M.: What is SMDI?, Electronic Musician 9(6). P.7982
  - Página Web:
    Sun Microsystems: The Java Tutorial. http://java.sun.com. Noviembre 2000.

Usted deberá entregar un correo electrónico debidamente identificado (nombre y carnet de los integrantes del equipo y nombre del proyecto):

- Una carpeta que contenga todos los archivos de Python que utilizaron
- Un link al repositorio en github
- Un informe según la estructura sugerida anteriormente.
- El listado del programa debidamente documentado.

La entrega debe realizarse a la fecha discutida en el laboratorio (esta fecha aún está en discusión).

Importante: el proyecto es un trabajo en equipos de a dos personas, **NO** puede haber ningún tipo de intercambio con otros grupos.