Trabajo Práctico Especial
Programación Imperativa
Segundo Cuatrimestre 2009

1. Objetivo.

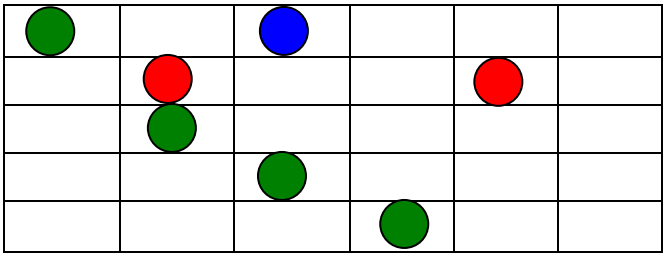
Diseñar e implementar una variante del juego "Colored lines" (http://www.gamesforthebrain.com/game/colorlines/).

2. Descripción del juego.

La versión original es un juego de un único jugador que consta de un tablero rectangular que inicialmente cuenta con un número prefijado de fichas (por ej. 5) de distintos colores esparcidas al azar.

En cada turno el jugador puede mover una bolilla a una posición siempre y cuando la misma esté libre y haya un camino para llegar al destino. Un camino está compuesto por una serie indeterminada de pasos, y un paso consiste en desplazarse hacia una celda vacía hacia abajo, la derecha, izquierda o arriba (no se puede mover en diagonal).

Cada posición está indicada por una coordenada [fila, columna], donde la coordenada superior izquierda es la [0,0]. En el siguiente tablero, por ejemplo, es posible desplazar la bolilla verde que se encuentra en [0,0] únicamente a [0,1], [1,0], [2,0], [3,0], [3,1], [4,0], [4,1], [4,2]

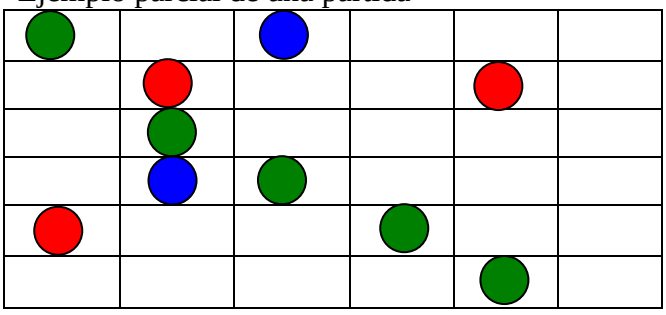


Una vez que el usuario hizo su movida pueden pasar dos cosas:

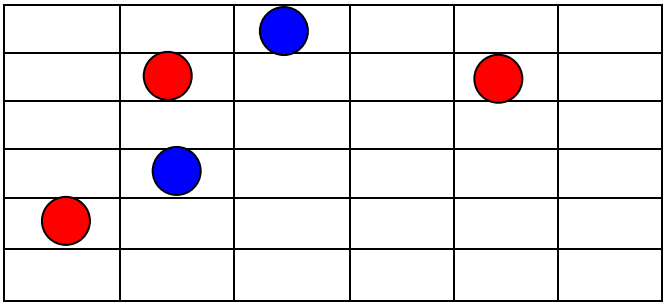
- Si alineó 5 o más bolillas del mismo color, ya sea en forma vertical, horizontal o diagonal, esas bolillas desaparecen y al usuario se le otorga un puntaje de acuerdo a la cantidad de bolillas que alineó.
- Si no alineó 5 bolillas se colocan al azar una cantidad prefijada de nuevas bolillas (por ej. 3), cuyos colores también son elegidos al azar. Si una de las bolillas que se coloca al azar forma una línea de 5 o más bolillas del mismo color, las mismas se retiran del tablero. En este caso no se computa el puntaje para el usuario.

y luego el usuario sigue moviendo.
La partida termina cuando no quedan celdas vacías.

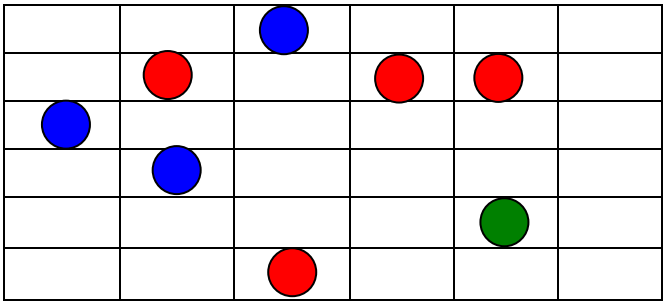
Ejemplo parcial de una partida



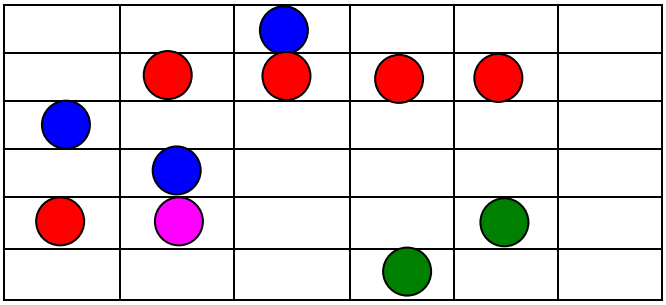
Acción: [0,0] -> [1,0]



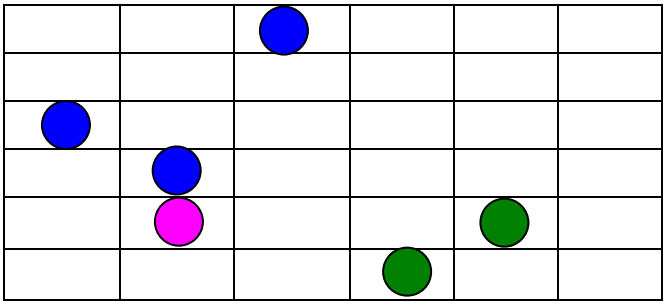
Acción: [4,0] -> [1,3]



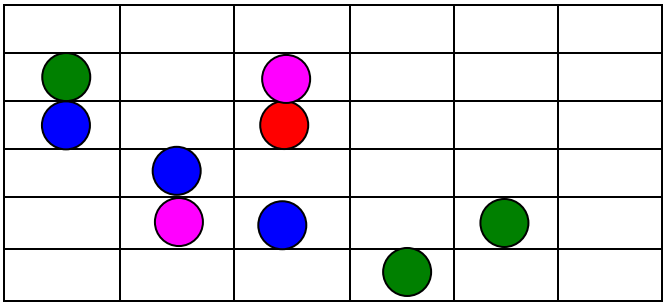
Acción: [5,2] -> [1,2]



Acción: [4,0] -> [1,5]



Acción: [0, 2] -> [4,2]



etc.

3. Descripción funcional del programa.

El programa deberá presentar un menú que permita elegir entre las siguientes opciones:

1. Juego de un jugador modalidad normal.
2. Juego de un jugador modalidad por tiempo *(deberá solicitar la duración en minutos)*
3. Juego de dos jugadores en modalidad normal
4. Recuperar un juego grabado *(deberá solicitar el nombre del archivo)*
5. Terminar.

Una vez elegida la opción, si la misma corresponde a un nuevo juego, el usuario deberá especificar:

- Dimensiones del tablero (mínimo 5x5). No hace falta que el tablero sea cuadrado.
- Cantidad de colores a utilizar (mínimo 2, máximo 9)
- Cantidad de fichas que hay inicialmente en el tablero
- Cantidad de bolillas que forman una línea
- Cuántas bolillas se agregan en cada turno

Terminadas de establecer las opciones, el o los tableros se completarán en forma aleatoria. Para representar los distintos colores se utilizarán los dígitos del 1 al 9.

Se le mostrará al usuario el tablero y, si corresponde, también el de su adversario, y se le pedirá que ingrese su jugada con el siguiente formato:

- **[F1, C1] [F2,C2]**

que representa una jugada para mover la ficha de la coordenada [F1, C1] a [F2,C2]. En caso de ser inválido se le muestra un cartel aclarando por qué es inválido y se le vuelve a solicitar el comando.

Además, el usuario podrá ingresar los siguientes comandos:

- **save *filename*** guardará el juego en un archivo de nombre *filename*.
- **quit**: saldrá del juego.
- **undo**: volverá al estado anterior. En caso de repetir en forma inmediata esta acción, la segunda no tendrá efecto.

El comando undo es válido únicamente si hay un solo jugador, no se puede deshacer si son dos jugadores.

En caso de haberse ingresado una jugada con coordenadas válidas se mostrará el nuevo tablero, la cantidad de puntos hechos en esa jugada y la cantidad de puntos acumulados para ambos jugadores.

Si un jugador no tiene celdas libres sigue jugando su adversario hasta que no pueda mover.

3.1. Cálculo del puntaje

El puntaje de cada jugada dependerá de la cantidad de fichas eliminadas en forma simultánea:

Elimina una línea con la mínima cantidad: 1 punto
Elimina una línea con una ficha más: 2 puntos
Elimina una línea con dos fichas más: 4 puntos
Elimina una línea con tres fichas más: 6 puntos
En cualquier otro caso: 8 puntos

3.2. Determinación del ganador.

Si es un juego de un jugador, en modo común, el mismo finalizará cuando no haya celdas libres.

Si es un juego de un jugador, en modo por tiempo, el mismo finalizará cuando se haya vencido el tiempo o no queden celdas libres, lo que ocurra primero.

En caso de haber elegido jugar contra un adversario, si uno de los dos no puede mover, sigue jugando el otro hasta que no pueda seguir más. Una vez que ambos no puedan elegir posición, se determinará quién fue el ganador en base al puntaje.

4. Diseño e Implementación del programa.

Se debe realizar un diseño donde se separe claramente la interfaz con el usuario (front-end) del procesamiento de los datos del juego (back-end). Esto se verá reflejado en la implementación de una biblioteca de funciones de back-end en el archivo `colorsBack.c` y otro conjunto de funciones de front-end que invocan a la biblioteca. Estas últimas incluyen a la función main y corresponden al archivo `colorsFront.c.`

En ningún caso se debe repetir código para resolver situaciones similares, sino que debe implementarse una correcta modularización y se deben reutilizar funciones parametrizadas.

Tanto la biblioteca como el front-end deben estar correctamente comentados y en el caso de la biblioteca se debe escribir el archivo de encabezado correspondiente.

El formato del archivo en el cual se almacena un partido debe ser el siguiente (se debe respetar estrictamente el orden y tipo):

- Un entero con el valor cero si el juego es de un jugador en modo normal, uno si es por tiempo, y dos si es de dos jugadores.
- Si era en modo tiempo, un entero con la cantidad de segundos que restan del juego
- Si era de dos jugadores, un entero con el número 1 si el próximo turno le corresponde al jugador 1 ó un 2 si el próximo turno es para el jugador 2.
- Un entero con la cantidad de filas del tablero
- Un entero con la cantidad de columnas del tablero
- Un entero con la cantidad de colores que se usaron.
- Un entero con la cantidad de fichas necesarias para hacer línea
- Un entero con la cantidad de fichas que se agregan en cada turno
- Un entero con los puntos hechos hasta el momento por el jugador

Una secuencia de chars que corresponde a cada una de las posiciones del tablero del usuario, ordenados por fila. La celda superior izquierda es la que debe aparecer primero, seguida del resto de la fila. Luego la segunda fila y así sucesivamente. Los valores de cada celda son:

- ‘0’: Posición vacía
- ‘1’: Color 1
- ‘2’: Color 2
- ....
- ‘n’: color N

Si es un juego de dos jugadores, además el archivo contendrá:

- Un entero con los puntos hechos por el jugador 2
- Una secuencia de chars que corresponde a cada una de las posiciones del tablero del jugador 2, en forma análoga al tablero del usuario

El archivo del partido guardado **no** debe ser editable con un editor de texto, su formato es “binario”.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado.

ALGUNOS CONSEJOS:

No escribir el programa entero y después probarlo todo junto, sino escribir cada función, probándola por separado. Programar defensivamente en todos los casos.

Escribir el esquema de cada función primero en papel, pudiendo utilizar pseudocódigo o lenguaje coloquial.

Una buena metodología es comenzar a escribir las funciones de más alto nivel, *cableando* las inferiores con *cuerpo nulo* o con un *valor fijo*, e ir reemplazándolas de a una por vez, en la medida en que al integrarlas todo siga funcionando correctamente. Esto es implementación **Top-Down**.

Otra metodología es comenzar a escribir las funciones desde el nivel inferior, una por vez, probando a cada una con un pequeño main que sólo la invoque a ella. Una vez escritas y verificadas todas las funciones, unificarlas en un solo módulo. Esto es implementación **Bottom-Up.**

5. Material a entregar.

Cada grupo deberá entregar en sobre manila, con el nombre de los integrantes en el frente, el siguiente material:

- Impresión de todos los códigos fuente.
- Impresión del árbol de funciones (como los presentados en el TP Nro. 7)

A su vez, cada grupo deberá enviar un archivo tipo zip a la cuenta `pi@it.itba.edu.ar` con los siguientes archivos:

- `colorsFront.c`
- `colorsBack.h`
- `colorsBack.c`
- `makefile`
- Otros programas fuentes (de ser necesario)
- Ejemplos de partidas almacenadas.

La impresión de los códigos debe hacerse de forma tal que el código pueda leerse fácilmente. Ejemplos a no seguir:

Código mal indentado

```
for(i=0; i < filas; i++)
for(j=0; i < columnas; i++)
count = …
```

Líneas que ocupan más del ancho de página y continúan en otra línea

```
while ( … )
{
     for ( i = 0; i < filas; i ++ )  /* en este ciclo vamos a
verificar si hay posiciones libres */
     {
          .....
```

Comentarios innecesarios

```
i++; // Incrementamos la variable i en 1
```

6. Fecha de entrega.

El trabajo debe entregarse por e-mail el 20 de noviembre hasta las 20 hs (todos los mails recibirán confirmación de lectura), los materiales impresos se deben entregar el viernes 20 de noviembre en el horario de clases.

Aquel grupo que entregue el trabajo **en fecha** y resulte desaprobado, tendrá la oportunidad de recuperarlo. Si el recuperatorio resultara satisfactorio, la nota del mismo será de 4 puntos.

Por otra parte, los alumnos pueden optar por entregar tarde (hasta una semana después de la fecha prevista), pero en ese caso **se le restarán dos puntos** a la nota del trabajo y **no tendrán posibilidad de recuperatorio**.

7. Criterios de Evaluación y Calificación

Para la evaluación y calificación del trabajo especial se considerarán:

- el correcto funcionamiento del programa (recordar el uso de métodos de prueba de software)
- la modularización realizada
- la claridad del código
- el cumplimiento de las reglas de estilo de programación dadas en clase
- la presentación del trabajo en todos los aspectos (documentación, facilidad de uso, etc

8. Consultas

Cualquier consulta que se desee realizar sobre el enunciado o la implementación del trabajo especial deberá hacerse en el horario propio de consulta de la materia.