Trabajo Práctico Especial
Programación Imperativa
Primer Cuatrimestre 2014

"Minoku"

Objetivo

Implementar el juego de tablero llamado "Minoku" consiste en detectar las minas de un tablero de NxM en base a la información provista por el mismo.

Descripción del Juego

El juego consiste en lograr detectar todos los casilleros que tienen minas dentro de un tablero en base a la información provista. De un tablero NxM se sabe cuántas minas contiene una fila y una columna. La cantidad de minas será provista consultando a cada fila o columna en forma de una secuencia de aparición de las mismas

Tablero

El tablero es un rectángulo de F filas y C columnas, cuyo tamaño mínimo será de 5x5, sin dimensión máxima, pudiendo ser esta NxM (n,m >=5). Dadas las limitaciones de la consola el front end soportará hasta 19x19.
En el tablero las minas estarán distribuidas de forma aleatoria en base a la dificultad del mismo.
De un tablero se conoce la distribución de las minas en base a su aparición por filas y columnas. Se puede consultar a una fila o columna determinada un número ilimitado de veces.
Se disponen de N banderines, estos sirven al jugador como señaladores de los casilleros donde podría encontrarse una mina.

Ítems del tablero

- **Banderín**: se disponen de N cantidad de banderines, donde N es la cantidad de minas. Estos estarán representados por el carácter & en el tablero
- **Minas**: N minas distribuidas aleatoriamente por el tablero. Estas estarán representadas por el carácter #, solo estarán visibles cuando el jugador finalice el nivel.

Dificultad de los tableros

- **1 Fácil:** la cantidad de minas representa un 20% del total del tablero, 10 undo
- **2 Medio:** la cantidad de minas representa un 50% del total del tablero, 5 undo
- **3 Difícil:** la cantidad de minas representa un 70% del total del tablero, 3 undo
- **4 Pesadilla:** la cantidad de minas representa un 90% del total del tablero (solo para tableros de más de 100 casillas), 1 undo

Disposición inicial de las fichas.

Al inicio de la partida, el tablero se encuentra en blanco. Se le muestran al jugador la cantidad de banderines disponibles, movimientos restantes y undo restantes.

	1	2	3	4	5	6	7	8	9
A									
B									
C									
D									
E									
F									

Tablero real (oculto)

	1`	2	3	4	5	6	7	8	9
A	#								#
B		#	#	#		#		#	
C		#	#						
D			#	#	#				#
E		#	#						
F				#					

Fin del nivel.
El juego termina cuando ocurre alguna de las siguientes situaciones:

- se verifica que todas las minas han sido detectadas con banderines correctamente ubicados(gana)
- todos los casilleros en blanco fueron descubiertos(gana)
- el jugador se topa con una mina sin posibilidad de deshacer la jugada (undo)
- el jugador excede la cantidad de movimientos permitidos, en caso de jugarse con límite de movimientos. El programa deberá detectar las condiciones en las cuales el nivel ya no podrá ganarse. Es decir, si por más que ejecute todo el número de movimientos disponibles, ya no podrá descubrir todas las bombas

En cualquier caso, al finalizar el juego debe indicarse el resultado: GANO o PERDIO.

Descripción funcional del programa

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

1. Juego nuevo.
   1.1. Juego individual sin límite de movimientos.

1.2. Juego individual con límite de movimientos.

1.3. Juego por campaña (siempre es con límite de movimientos)

2. Recuperar un juego grabado *(deberá solicitar el nombre del archivo)*
3. Terminar.

1. Juego Nuevo

1.1. Juego Individual sin límite de movimientos.

Al comenzar un juego nuevo el usuario deberá especificar **las dimensiones** del tablero (mínimo 5x5, máximo 19x19). Indicará número de filas y número de columnas.

Luego, deberá elegir el nivel.

Una vez establecidas las dimensiones, se ubicarán (aunque no a la vista del jugador) las fichas de manera automática, según lo indicado en los apartados "**Disposición inicial de las Fichas**", "**Dificultad de los tableros**" e "**Ítems del tablero"**.

El tablero al inicio se mostrará en blanco al jugador, es decir con todos sus casilleros con '0'.

Ejemplo:
Dimensión del tablero: 6x9
Nivel: 2
Número de banderines disponibles: 27 (hay 27 bombas)
Número de undo disponibles: 5
Las filas se indican con letras (A-Z) y las columnas con números (1-19)

El tablero que se le muestra al jugador es:

	1	2	3	4	5	6	7	8	9
A									
B									
C									
D									
E									
F									

El tablero **real oculto** es:

	1	2	3	4	5	6	7	8	9
A	#								#
B	#	#	#	#		#		#	
C		#	#			#		#	#
D	#		#	#	#				#
E	#	#	#					#	
F		#		#		#		#	#

Las jugadas ingresadas pueden ser:

- **S** (F,C)
  barrer el casillero (si hay una mina pierde)
- **unflag/flag** (F,C)
  sacar/poner un banderín en el casillero F,C
- **unflag/flag** (FO,CO:FD,CD)
  sacar/poner una serie de banderines desde el casillero origen FO,CO hasta el casillero FD,CD. No se permiten diagonales. Solo desplazamientos sobre columnas o filas. Debe controlar que disponga de suficientes banderines. Debe validar el rango y que la operación de unflag pueda concretarse.
- **query** F o **query** C:
  devuelve la secuencia de aparición de las minas en esa determinada fila o columna. F es una letra, C es un número.
- **save** *filename*
  guardará el juego en un archivo de nombre *filename*.
- **quit**
  saldrá del juego, preguntando si desea guardar el mismo en un archivo.
- **undo**
  volverá al estado anterior. En caso de repetir en forma inmediata esta acción, la segunda no tendrá efecto. Debe controlar que tenga disponibles la cantidad de undo. Llegado el caso de que el jugador se tope con una mina, y disponga de comandos undo, no se le debe mostrar el tablero real, sino darle la posibilidad de deshacer la jugada o darse por vencido.

En caso de haberse ingresado una jugada con coordenadas válidas, se mostrará:

1. Los banderines restantes.
2. La cantidad de undo disponibles.
3. El nuevo tablero.

Para representar el banderín se utilizará el carácter '**&**'

Para representar los casilleros vacíos que fueron correctamente descubiertos se utilizará el carácter '-'. Recordar que si un casillero se elige con el comando S(f,c), y ahí hay una mina, y no se tienen **undo** disponibles, se pierde inmediatamente. Por lo que sólo se mostrarán con '-' aquellos casilleros que fueron descubiertos como vacíos y realmente lo están.

Para representar las minas se utilizará, internamente el carácter '**#', pero esto sólo se mostrará al final del juego.**

En el ejemplo:

query 1
2-2
query A
1-1
query F
1-1-2
flag(A,1:B,1)

	1	2	3	4	5	6	7	8	9
A	&								
B	&								
C									
D									
E									
F									

Banderines restantes: 25
Undo restantes: 5

El tablero **real oculto** es:

	1	2	3	4	5	6	7	8	9
A	&								#
B	&	#	#	#		#		#	
C		#	#			#		#	#
D	#		#	#	#				#
E	#	#	#					#	
F		#		#		#		#	#

s(B,2)
undo
s(A,2)

	1	2	3	4	5	6	7	8	9
A	&	-							
B	&								
C									
D									
E									
F									

Banderines restantes: 25
Undo restantes: 4

Etc.

1.2. Juego Individual con límite de movimientos.

Al comenzar un juego nuevo el usuario deberá especificar **las dimensiones** del tablero (mínimo 5x5, máximo 19x19). Indicará número de filas y número de columnas.

Luego, deberá elegir el nivel.

En función del nivel se calcula el límite de movimientos como:

Límite de movimientos = número de minas + número de undos.

Se considera movimiento a cada uno de los siguientes comandos:

- **S** (F,C)
- **unflag/flag** (F,C)
- **unflag/flag** (FO,CO:FD,CD)
- **undo**

En caso de haberse ingresado una jugada con coordenadas válidas, se mostrará:

1. Los banderines restantes.
2. La cantidad de undo disponibles.
3. La cantidad de movimientos disponibles.
4. El nuevo tablero.

En el ejemplo anterior:

Dimensión del tablero: 6x9
Nivel: 2
Número de banderines disponibles: 27 (hay 27 bombas)
Número de undo disponibles: 5
Número de movimientos disponibles: 32
query 1
2-2
query A
1-1
query F
1-1-2
flag(A,1:B,1)
Banderines restantes: 25
Undo restantes: 5
Movimientos restantes: 31
s(B,2)
undo
s(A,2)
Banderines restantes: 25
Undo restantes: 4
Movimientos restantes: 28

1.3.Juego por campaña

La campaña es una serie de niveles, almacenados en un archivo de texto.
El archivo de texto tiene la forma:
DIFICULTAD DIMENSION
DIFICULTAD DIMENSION
…
DIFICULTAD DIMENSION [EOF]

(el número de dificultad y dimensión se separan con una tabulación)
Ejemplo:
campañaSimple.txt

1 5x5 1 7x5 1 7x7 2 5x5

El juego comienza pidiendo este archivo de texto.
Se juega igual que como el juego individual, pero se da por finalizado cuando se completan todos los niveles del archivo de campaña.

2. Recuperar juego

2.1.Juego individual guardado

El archivo debió ser guardado durante alguna partida con la opción **save *filename*,** o bien al hacer **quit** y decidir guardar.

El archivo **binario** en el cual se almacena un partido debe ser el siguiente (respetando el orden y tipo)

- Un entero con el nivel.

- Un entero con la cantidad de filas del tablero.
- Un entero con la cantidad de columnas del tablero.
- Un entero con el valor 0 si se juega sin limite de movimientos, o con el valor correspondiente al número de movimientos disponibles si se juega con límite.
- Un entero con el valor 0 si es un juego individual, o el valor 1 si es un juego por campaña.

Una secuencia de chars que corresponde a cada una de las posiciones del tablero real, 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:

- '-': Posición vacía
- '#': Posición con bomba

Una secuencia de chars que corresponde a cada una de las posiciones del tablero visible al jugador, 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 no tocada (no hay banderín ni se ha descubierto celda vacía)
- '-': Posición vacía descubierta correctamente.
- '&': Posición con banderín.

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

2.2.Juego de campaña guardado

Si el juego es por campaña, al final del archivo binario debe haber un string terminado en '.txt' con el nombre del archivo de campaña.

Cuando se desea recuperar un juego de campaña guardado se debe abrir dos archivos:

- El archivo de juego (binario)
- El archivo de campaña (archivo de texto)

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 `minokuBack.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 `minokuFront.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 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.**

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:

- `minokuFront.c`
- `minokuBack.h`
- `minokuBack.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
```

Fecha de Entrega

El trabajo debe entregarse por e-mail a pi@it.itba.edu.ar. La entrega será por mail el 9 de junio antes de las 21 hs. Los materiales impresos se entregarán al comienzo de la clase del martes 10 de junio en el horario de clases.
La fecha de entrega tardía es el 16 de junio, antes de las 21hs. Recordar la política de entrega tardía detallada en el reglamento de la materia.

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 que opten por entregar en fecha tardía **se le restarán dos puntos** a la nota del trabajo y **no tendrán posibilidad de recuperatorio**.

Los grupos deberán tener un máximo de 3 integrantes.
Un representante del grupo deberá enviar un e-mail a la cátedra a pi@it.itba.edu.ar antes del 20 de mayo a las 21 hs registrando la existencia del grupo y especificando los integrantes del mismo. Se considerará que aquellos alumnos que no pertenezcan a un grupo registrado han optado por no realizar el Trabajo Práctico Especial.

Criterios de evaluación y calificación

Para la evaluación y calificación del trabajo especial se considerarán:

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