Laboratorio de Programación Avanzada
PROGRAMACIÓN FUNCIONAL – Práctica de evaluación Septiembre 2010

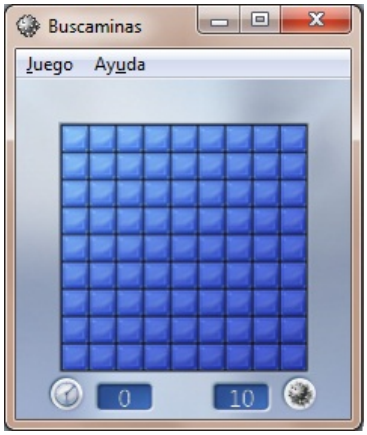
ENUNCIADO.

Publicado el 23 de Junio de 2010

Buscaminas

El Buscaminas (en inglés: Minesweeper) es un videojuego para un jugador inventado por Robert Donner en 1989. El objetivo del juego es despejar un campo de minas sin destapar ninguna mina.

El juego ha sido programado para muchos sistemas operativos, pero debe su popularidad por la versión que viene con Microsoft Windows.



Tablero 1

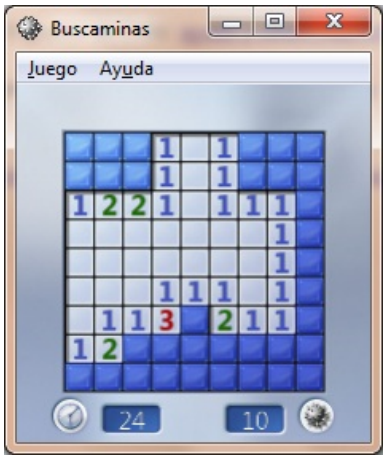
El juego consiste en despejar todas las casillas del tablero que no oculten una mina.

El tablero y el número de minas depende del nivel de juego. Los niveles, para la nueva versión de Windows 7, son:

- Nivel principiante: 9 × 9 casillas y 10 minas.
- Nivel intermedio: 16 × 16 casillas y 40 minas.
- Nivel experto: 16 × 30 casillas y 99 minas.
- Nivel personalizado: en este caso el usuario personaliza su juego eligiendo el número de minas y el tamaño del tablero.

Para el programa que vamos a realizar, solamente consideraremos el nivel principiante (Tablero 1).

Aunque en el programa visual de Windows, las casillas se seleccionan "pinchando" en ellas con el ratón, en nuestro programa las seleccionaremos refiriéndonos a ellas mediante un número de dos cifras que representa la fila y la columna. Por ejemplo, la casilla 34 es la que está situada en la fila 3 columna 4.



Tablero 2

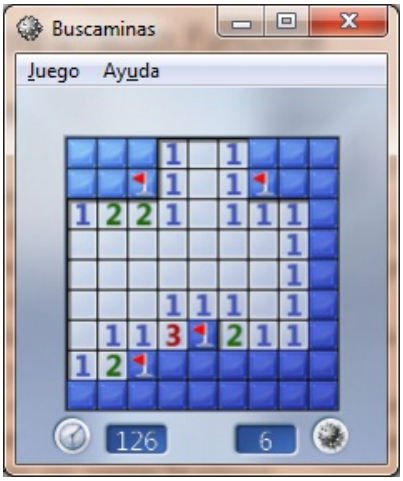
Reglas

a) Las casillas descubiertas, si no tienen una mina, se muestran vacías o con un número. El número indica las minas que suman todas las casillas circundantes. Así si una casilla tiene el número 3 significa que de las ocho casillas que hay alrededor (si no está en una esquina o borde) hay 3 con minas y 5 sin minas.

b) En caso de que la casilla descubierta no tuviese ninguna mina vecina, el programa descubrirá todas las casillas adyacentes y si entre ellas volviese a darse la condición de que hay casillas sin minas vecinas repite el proceso hasta que en todas aparezca un número de minas vecinas distinto de 0 (Tablero 2).

c) Si se descubre una casilla con una mina se pierde la partida.

d) Se puede poner una marca en las casillas que el jugador piensa que hay minas para ayudar a descubrir la que están cerca (Tablero 3).



Tablero 3

e) El juego termina cuando todas las casillas libres de minas han sido descubiertas (Tablero 4 - juego ganado) o cuando se ha seleccionado una casilla que contenía una mina (Tablero 5 - juego perdido).

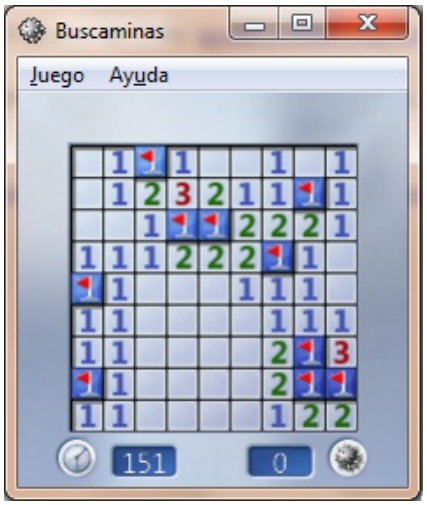
Entrenamiento

Se puede ver el funcionamiento en cualquier versión del sistema operativo Windows o en alguna de las páginas web que ofrecen juegos interactivos de este tipo, como:

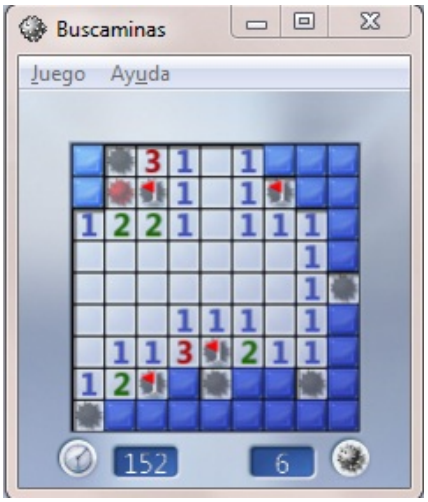
http://www.buscaminas.org/

o

http://club.telepolis.com/jagar1/Buscaminas/Buscaminas.htm



Tablero 4



Tablero 5

1.- Programación del juego “Buscaminas”

Queremos crear un archivo ejecutable, programado en Caml, que simule el comportamiento del juego en un entorno no gráfico, con tableros de 10 x 10 casillas y 16 minas colocadas aleatoriamente en cada partida. Para ello, escribiremos las funciones necesarias para ir modificando el tablero convenientemente, de acuerdo con las vicisitudes del juego.

Como no se va a disponer de un ratón con el que seleccionar una casilla del tablero, se propone denominar cada casilla mediante un número de dos cifras que indique el número de fila (del 0 al 9) y el número de columna (del 0 al 9). Así, por ejemplo, la casilla donde se encuentra el primer número 3 del tablero 4, sería la 13 (fila 1, columna 3).

Para permitir las pruebas del programa, se mostrará, al comienzo de cada partida, el tablero generado, con la situación de todas las minas, de forma que podamos realizar las jugadas que queramos, para comprobar su funcionamiento.

2.- Representación del tablero del “Buscaminas”

Para gestionar la partida, el programa necesita un tablero con dos partes, una oculta, con las minas generadas al azar, y otra externa, para mostrar, con la situación del juego en cada momento. Para representar ambas partes, se utilizarán listas, una de enteros y otra de caracteres.

Para la parte oculta del tablero, como cada casilla puede tener entre 0 y un máximo de 8 minas a su alrededor, se utilizará una lista de enteros formada por números entre 0 y 8 para las casillas vacías y el 9 se reservará para las casillas con mina.

Para la parte visible del tablero, se usará una lista de caracteres con los siguientes valores: para las casillas todavía no descubiertas el carácter **`#`**; para las casillas ya descubiertas los caracteres **`.`, `1`, `2`, …`9`** según el valor 0 a 9 que tuviesen en la parte oculta; y para las casillas marcadas por el usuario como que tienen mina, el carácter **`X`.** (Si aparece un carácter `9` es porque se ha descubierto una casilla con mina y por tanto el juego ya no continúa).

(*NOTA: No se permite el uso de estructuras de datos modificables (vectores) para la representación del juego. Es necesario hacerlo mediante estructuras puramente funcionales (listas).).*

La primera función que tenemos que definir será:

```
iniciarTablero();;
```

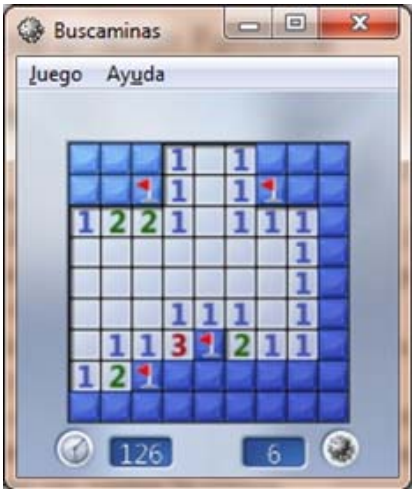
No toma argumentos y devuelve un tablero de juego formado por una tupla con las dos listas – parte oculta y parte visible - del Buscaminas con la estructura de datos elegida. Las posiciones de las minas se elegirán al azar cada vez que se ejecute la función.

3.- Representación del tablero de juego:

Necesitamos definir una función que nos permita mostrar el contenido del tablero del juego con el que estemos trabajando. Esta función **`mostrarTablero`** será la que nos permita ir mostrando la parte externa del tablero cada vez que el usuario realice un movimiento. Su funcionamiento sería como en el ejemplo siguiente, donde **tablero** es el tablero que representaría al que se acompaña en el gráfico adjunto.

```
mostrarTablero tablero;;
```

```
   0 1 2 3 4 5 6 7 8 9
  --------------------
0 |# # # 1 . 1 # # # #|
1 |# # # 1 . 1 # # # #|
2 |# # X 1 . 1 X # # #|
3 |1 2 2 1 . 1 1 1 # #|
4 |. . . . . . . 1 # #|
5 |. . . . . . . 1 # #|
6 |. . . 1 1 1 . 1 # #|
7 |. 1 1 3 X 2 1 1 # #|
8 |1 2 X # # # # # # #|
9 |# # # # # # # # # #|
  --------------------
- : unit = ()
```

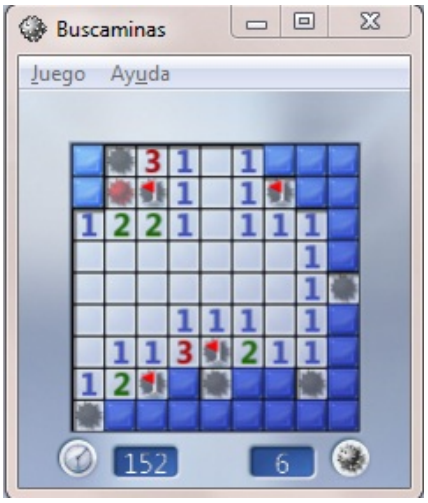


Tablero 3

También necesitamos otra función, que se ejecutará una sola vez al comienzo de cada juego, para mostrar el tablero inicial oculto generado por el programa (no se ejecutaría en un juego normal, solamente es para comprobar el funcionamiento del programa).

```
mostrarTabOculto tablero;;
```

```
   0 1 2 3 4 5 6 7 8 9
  --------------------
0 |2 9 3 1 . 1 1 1 1 .|
1 |2 9 3 1 . 1 1 1 1 .|
2 |2 9 9 1 . 1 9 1 . .|
3 |1 2 2 1 . 1 1 1 . .|
4 |. . . . . . . 1 1 1|
5 |. . . . . . . 1 9 1|
6 |. . . 1 1 1 . 1 1 1|
7 |. 1 1 3 9 2 1 1 1 .|
8 |1 2 9 3 9 2 1 9 1 .|
9 |9 2 1 2 1 1 1 1 1 .|
  --------------------
- : unit = ()
```



Tablero 5

4.- Funciones para el manejo del tablero:

Se podrán escribir todas las funciones que se consideren necesarias para iniciar un juego, permitir jugar al usuario definiendo qué quiere hacer (descubrir o marcar casillas), hasta que tropiece con una mina (juego perdido) o haya descubierto las 84 casillas que no tienen mina. En ambos casos, dará un mensaje de "lo siento ha perdido" o de "enhorabuena ha ganado" al usuario y terminará.

Entre todas las funciones que haya que definir, se pide escribir las funciones siguientes (con los nombres y sintaxis que aquí se muestran):

`contenido coord tablero;;`

Devuelve el contenido de la casilla coord en la parte oculta del tablero.

`vecinas coord;;`

Devuelve una lista con las coordenadas de las casillas que son vecinas de coord en un tablero de 10x10 casillas.

`marcar coord tablero;;`

Si la casilla no está descubierta, camba el `#` por una `X`. Si está ya descubierta no hace nada.

`descubrir coord tablero;;`

Cambia la casilla coord de la parte visible del tablero, de acuerdo con su contenido en la parte oculta. Si esa casilla está ya descubierta (no es `#` ni `X`) no hace nada.

1 – 8 : muestra ese valor en la parte visible.

9 : Se ha elegido una casilla con mina. En este caso se descubren todas las casillas que contienen una mina, para que el usuario las pueda ver.

0 : Se muestra un `.` en la casilla correspondiente de la parte visible y se descubren también todas las casillas vecinas que contengan otro 0. Se repite el proceso con las vecinas de éstas, hasta que no queden vecinas con 0 sin descubrir.

`completo tablero;;`

Detecta el final del juego devolviendo "cierto" en si se produce:

a) No queda ninguna casilla del tablero, que no contenga una mina, sin descubrir. En este caso imprime el mensaje: "Enhorabuena. Has ganado"
b) Todas las minas están descubiertas. En este caso imprime el mensaje: "Lo siento. Has perdido".

En ambos casos, el juego ya no continúa más.

`jugar jugadas tablero;;`

Muestra el tablero visible y escribe un mensaje solicitando una entrada, que tiene que ser una letra y dos dígitos (por ejemplo D32, M45, M77, D09). La letra puede ser D (descubrir) o M (marcar); el primer dígito es la fila y el segundo la columna de la casilla designada.
Solo se permiten esas dos letras y coordenadas válidas. Si la entrada no es correcta, (por ejemplo DM72, F43, H50, etc.) muestra el mensaje "Entrada inválida" y vuelve a empezar.
Si la entrada es válida, ejecuta la función "descubrir" o "marcar" según corresponda.
Esta función se ejecuta recursivamente hasta que el tablero esté completo (final del juego).
El parámetro "jugadas" irá contando el número de entradas que ha hecho el usuario. Se iniciará a 0 y cuando la partida termine se le comunicará al usuario cuántas jugadas ha hecho.

5.- Modo de funcionamiento:

Al arrancar el programa, se creará un tablero nuevo con las dos listas de 100 elementos correctamente rellenados. A continuación se mostrará la parte oculta del tablero y después se ejecutará la junción jugar, que tras mostrar la parte visible, imprimirá el mensaje.

```
Indique acción (D/M) y coordenadas de la casilla (por ejemplo D45):
```

Entonces el jugador deberá introducir una cadena válida con un carácter y dos dígitos.

En cada jugada, se irá realizando la acción solicitada sobre el tablero, se mostrará de nuevo el tablero con la nueva situación creada y se repite mensaje anterior.

Cuando el tablero se haya completado, porque no quedan más casillas sin minas sin descubrir, o porque todas las casillas con mina están al descubierto, se verá el mensaje correspondiente y el juego ha terminado.

La función **`jugar jugadas tablero`** debe llevar la cuenta del número de jugadas realizadas. La partida se inicia con el valor de jugadas=0 y se va incrementando en 1 con cada jugada (D o M). Al finalizar la partida, junto con el mensaje de "has ganado" o "has perdido", se mostrará el número de jugadas efectuadas.

El grupo de diseño del juego podrá introducir otras mejoras, de funcionamiento o de diseño, que considere interesantes.

6.- Creación del ejecutable:

Para terminar, compilar el programa CAML y crear el fichero **`Buscaminas.exe`**. Comprobar su buen funcionamiento realizando toda clase de jugadas, a la vista del tablero oculto generado, y viendo que el tablero va evolucionando de acuerdo con las reglas del juego.

CONDICIONES DE ENTREGA

1. La práctica se realizará por estudiantes individuales o en grupos de 2 personas cómo máximo.
2. Es condición necesaria para aprobar la práctica, que todos los programas funcionen correctamente y de acuerdo a las especificaciones indicadas en los enunciados.
3. Deberá entregarse un CD-ROM debidamente etiquetado y conteniendo los ficheros:
   - **Buscaminas.ml**, con todas las definiciones de funciones utilizadas en la práctica, con los comentarios que aclaren cómo funcionan, de forma que puedan ser cargadas – todas de una vez - en una sesión de CAML, para comprobar su funcionamiento durante la entrevista de defensa de la práctica.
   - **Buscaminas.exe**, con el programa anterior compilado y listo para ser ejecutado.
   - **Buscaminas.doc** o **Buscamians.pdf**, con la memoria de la práctica, como se describe en el apartado siguiente.
4. La memoria será un documento impreso en el que, para cada uno de los apartados de que consta la práctica, se describa el análisis de alto nivel efectuado sobre el problema a resolver, todas las funciones auxiliares que se vayan definiendo y la función principal. En todos los casos, se incluirá el código CAML de cada función y el tipo resultante. De cada función se explicará su funcionamiento y además, cuando se considere conveniente - para una mejor comprensión del funcionamiento de alguna función en particular, - se incluirá algún ejemplo de la aplicación de esa función sobre algún argumento interesante.
5. En la portada de la memoria, se incluirá:

   Ingeniería Técnica en Informática de _____"

   Laboratorio de Programación Avanzada. Curso 2009/10 (Septiembre)

   Práctica de Programación Funcional: BUSCAMINAS

   DNI – Apellidos, Nombre

   DNI – Apellidos, Nombre *(si se hace entre 2)*
6. La práctica deberá entregarse, al profesor asignado el día del examen de Septiembre (8 de Septiembre). Cada profesor pondrá una hoja en la puerta del laboratorio NL4 para que el alumno pueda elegir la hora de la entrevista.
7. Durante la defensa de la práctica, el profesor podrá solicitar a los alumnos la **modificación de alguna parte del código o la codificación de una nueva funcionalidad**. La incapacidad para llevar a cabo dichas tareas por parte de los alumnos conllevará la calificación de SUSPENSO en la práctica independientemente de la calidad del programa y la memoria entregados.